TESIS – SM 142501

METODE KLASIFIKASI BERBASIS MULTI KERNEL DENGAN PEMBELAJARAN YANG BERTAMBAH UNTUK TEMU KEMBALI CITRA

MUHAMMAD ATHOILLAH
NRP 1213 201 056

DOSEN PEMBIMBING
Prof. Dr. H. M. Isa Irawan, M.T.
Dr. Elly Matul Imah, M. Kom.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS – SM 142501

MULTI KERNEL-BASED CLASSIFICATION METHOD WITH INCREMENTAL LEARNING FOR IMAGE RETRIEVAL

MUHAMMAD ATHOILLAH
NRP 1213 201 056

SUPERVISOR
Prof. Dr. H. M. Isa Irawan, M.T.
Dr. Elly Matul Imah, M. Kom.

MASTER'S DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2015

METODE KLASIFIKASI BERBASIS MULTI KERNEL DENGAN PEMBELAJARAN YANG BERTAMBAH UNTUK TEMU KEMBALI CITRA

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
MUHAMMAD ATHOILLAH
NRP. 1213 201 056

Tanggal Ujian : 10 Juli 2015
Periode Wisuda : September 2015

Disetujui oleh:

Prof. Dr. Mohammad Isa Irawan, M.T. (Pembimbing I)
NIP. 19631225 198903 1 001

Dr. Elly Matul Imah, S.Si., M. Kom. (Pembimbing II)
NIP. 19820405 200501 2 002

Dr. Budi Setiyono, S.Si., M.T. (Penguji)
NIP. 19720207 199702 1 001

Dr. Dwi Ratna Sulistyaningrum, S.Si., M.T. (Penguji)
NIP. 19690405 199403 2 003

Direktur Program Pascasarjana,

Prof. Dr. Ir. Adi Soeprijanto, M.T.
NIP. 19640405 199002 1 001

METODE KLASIFIKASI BERBASIS MULTI KERNEL DENGAN PEMBELAJARAN YANG BERTAMBAH UNTUK TEMU KEMBALI CITRA

Nama	:	Muhammad Athoillah
NRP	:	1213 201 056
Pembimbing	:	1. Prof. Dr. H. M. Isa Irawan, MT
		2. Dr. Elly Matul Imah, M. Kom.

ABSTRAK

Support vector machine (SVM) merupakan teknik yang mampu menyelesaikan masalah klasifikasi dengan baik. Walaupun begitu, ternyata untuk masalah domain yang bersifat nonlinier tidak bisa dipecahkan dengan SVM standar, sehingga perlu dimodifikasi dengan memasukkan fungsi kernel didalamnya. Ide dasar dari metode kernel ini adalah memodifikasi SVM dengan memetakan data *input* ke ruang vektor yang berdimensi lebih tinggi, sehingga pada ruang vektor yang baru ini, *hyperplane* dapat dikonstruksikan. Pada kenyataannya, menentukan fungsi kernel yang tepat untuk menyelesaikan klasifikasi yang dikerjakan dengan baik adalah hal yang sulit. Oleh karena itu, para peneliti mengembangkan pembelajaran kernel yang lebih fleksibel, yaitu dengan mengkombinasikan semua kernel selama proses pembelajaran yang biasa disebut dengan Pembelajaran Multi Kernel. Selain itu, hal penting lainnya dalam pembelajaran mesin adalah proses pembelajarannya, dimana pada umumnya sebuah sistem melakukan pembelajaran yang baru setiap penambahan kelas baru, sehingga seolah-olah pembelajaran yang lalu tidaklah berarti, hal ini tentunya tidaklah efektif. Pada penelitian ini dibangun klasifikasi dengan metode SVM yang berbasis multi kernel yang kemudian diaplikasikan pada aplikasi temu kembali citra (*image retrieval*) dengan teknik pembelajaran yang bertambah (*incremental*) artinya apabila terjadi penambahan data atau informasi baru tidak harus menghapus pengetahuan yang lalu dan mengulang pembelajarannya dari awal. Hasil yang didapat dari penelitian ini menunjukkan bahwa metode klasifikasi berbasis multi kernel dengan pembelajaran yang bertambah memiliki performa yang baik dengan rata-rata nilai *precision* mencapai 42%, *recall* 38% serta akurasi dengan nilai 87% lebih baik bila dibandingkan dengan pembelajaran standar dan berbasis kernel tunggal RBF dimana nilai *precision* hanya 3%, *recall* 12% serta akurasi 80%, atau Polynomial dengan nilai *precision* 40%, *recall* 35% serta akurasi 86%. Hasil penelitian juga menunjukkan bahwa dengan metode pembelajaran yang bertambah, waktu komputasi yang dibutuhkan selama proses *training* lebih cepat bila dibandingkan dengan metode pembelajaran yang standar. Dimana waktu *training* rata-rata yang dibutuhkan sistem mencapai 8.7334 lebih cepat 2.0755 detik bila dibandingkan dengan SVM Polynomial dan 4.0546 detik lebih cepat daripada SVM RBF.

Kata kunci : *Support Vector Machine*, Klasifikasi, Pembelajaran Multi kernel, Pembelajaran yang Bertambah, Temu Kembali Citra.

MULTI KERNEL-BASED CLASSIFICATION METHOD WITH INCREMENTAL LEARNING FOR IMAGE RETRIEVAL

Name	:	Muhammad Athoillah
Student Identity Number	:	1213 201 056
Supervisor	:	1. Prof. Dr. H. M. Isa Irawan, MT
		2. Dr. Elly Matul Imah, M. Kom.

ABSTRACT

Support Vector Machine is technique that can solve classification problem well. Although that, the regular SVM can't solve classification problems deal with nonlinear domain, to solving that one the regular SVM need to be modified by put Kernel Function to that SVM. The main idea of this kernel method is modified SVM that maping input data to the higher-dimensional space, so in this new space, the hyperplane can be constructed. However selecting the precise kernel to solve classification problem well is quite difficult. So the machine learning practitioner may be interested in more flexible models, that is combining multiple kernel during training process that we usually called this *Multiple Kernel Learning* (MKL) method. Beside that, another important thing in machine learning is learning process problem, which is the system usually do a new training process when a new class is added to it. This process is certainly not effective, because its mean that previous training process is in vain. The framework in this paper is to build a classifier with SVM MKL-based method and apply in image retrieval with incemental learning. It's mean that if new data or new information is added to the system, it doesn't has to be repeat the learning process since the learner can be simply updated. The result shows that Incremental Multiple Kernel Learning method has good perform with average of precision value reach 42%, recall 38% and accuracy 87% better than standart learning method with single kernel RBF which is the precision value is only 3%, recall 12% and accuracy 80%, or Polynomial kernel with precision value is 40%, recall 35% and accuracy 86%. The result also shows that incremental learning method has faster computation time during training session than standart learning method. Where the average training time of system is 8.7334 second, 2.0755 second faster than SVM Polynomial and 4.0546 second faster and SVM RBF.

Key Words : Support Vector Machine, Classification, Multiple Kernel Learning, Incremental Learning, Image Retrieval.

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, segala puji syukur penulis panjatkan kehadirat Allah SWT. Kemudian, shalawat dan salam semoga senantiasa tercurah atas Rasulullah SAW, bagi keluarga dan sahabat yang mulia serta pengikut Beliau yang setia hingga akhir zaman. Amin.

Berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan tesis yang berjudul **"METODE KLASIFIKASI BERBASIS MULTI KERNEL DENGAN PEMBELAJARAN YANG BERTAMBAH UNTUK TEMU KEMBALI CITRA"** sebagai salah satu persyaratan kelulusan dalam memperoleh gelar Master di Program Studi Magister Matematika, Fakultas MIPA, Institut Teknologi Sepuluh Nopember.

Kelancaran proses penulisan tesis ini berkat bimbingan, arahan, dan petunjuk dari berbagai pihak, baik moral maupun spiritual. Penulis dalam kesempatan ini menyampaikan rasa hormat dan ucapan terima kasih yang sebesar-besarnya kepada:

1. Ayahanda Yusup, Ibunda Lu'aiyah, Adinda Nuril dan Nadifah, atas segala dukungan dan doanya selama penulis menuntut ilmu magister matematika.
2. Bapak Prof. Dr. Mohammad Isa Irawan, M.T. dan Ibu Dr. Elly Matul Imah, M.Kom atas bimbingan, arahan, dan waktu yang telah diluangkan kepada penulis untuk berdiskusi selama menjadi dosen pembimbing dan perkuliahan.
3. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si., M.T yang telah memberikan masukan dan saran selama penulis menjalani perkuliahan, baik sebagai dosen wali maupun penguji pada saat seminar hasil dan sidang tesis.
4. Bapak Dr. Budi Setiyono, S.Si., M.T. selaku dosen penguji yang telah memberikan masukan guna menyempurnakan tesis ini.

5. Bapak Dr. Subiono, MS selaku Ketua Program Studi Pascasarjana Jurusan Matematika, dan Ibu Prof. Dr. Erna Apriliani, M.Si. selaku Ketua Jurusan yang telah memberikan dukungan dan kemudahan dalam penyelesaian tesis.
6. Direktorat Jenderal Pendidikan Tinggi selaku pemberi Beasiswa Fresh Graduate untuk studi pasca sarjana di Matematika ITS pada tahun 2013.
7. Bapak/Ibu dosen yang telah memberikan bekal dan ilmu pengetahuan, selama berada di matematika ITS, serta staf administrasi Program Studi Magister Matematika atas segala bantuannya.
8. Teman-teman Pascasarjana Matematika Angkatan 2013, terima kasih sudah menorehkan banyak kenangan dan berbagi ilmu selama masa perkuliahan.
9. Semua pihak yang tidak mungkin penulis sebutkan satu persatu yang telah banyak membantu sehingga tesis ini dapat terselesaikan.

Dengan keterbatasan pengalaman, pengetahuan maupun pustaka yang ditinjau, penulis menyadari bahwa tesis ini masih banyak kekurangan yang dibuat baik sengaja maupun tidak disengaja. Oleh sebab itu, penulis mengharapkan kritik dan saran agar tesis ini lebih sempurna serta sebagai masukan bagi penulis untuk penelitian dan penulisan karya ilmiah di masa yang akan datang.

Akhir kata, semoga tesis ini dapat bermanfaat bagi penulis sendiri, institusi pendidikan, dan masyarakat luas.

Wassalamualaikum Warahmatullahi Wabarakatuh

Surabaya, Juli 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pembelajaran mesin adalah satu disiplin ilmu dari *computer science* yang mempelajari bagaimana membangun komputer/mesin yang cerdas. Pembelajaran mesin dapat didefinisikan sebagai metode komputasional yang menggunakan pengalamannya untuk memperbaiki performa atau membuat prediksi yang akurat, pengalaman disini merujuk pada informasi masa lalu yang didapat dari perekaman data atau proses pengenalan yang dihasilkan dari analisa data (Mohri dkk, 2012). Pembelajaran mesin banyak digunakan untuk memecahkan masalah dalam hal pengklasifikasian, *clustering* dan prediksi. Karenanya, sampai saat ini pembelajaran mesin banyak diaplikasikan untuk membantu kebutuhan dan mempermudah kehidupan manusia diantaranya mesin kontrol dan robot, *game*, biologi komputasional, diagnosa medis, dll.

Pada dasarnya, kebanyakan dari algoritma pengklasifikasian yang telah dikembangkan dalam pembelajaran mesin selama ini mampu menyelesaikan masalah klasifikasi dengan baik, salah satu contohnya adalah adalah *Support Vector Machine* (SVM) (Rakotomamonjy dkk, 2008; Gosselin dkk, 2010; Athoillah dkk, 2015; Campbell & Ying, 2011; Tomasouw dkk, 2012). Sebuah algoritma dimana sistem pembelajarannya menggunakan ruang hipotesis berupa fungsi-fungsi linier dalam sebuah ruang fitur (*feature space*) berdimensi tinggi. Berbeda dengan strategi *neural network* yang berusaha mencari *hyperplane* (garis pemisah) antar kelas, SVM berusaha menemukan *hyperplane* yang terbaik pada *input space*, *hyperplane* pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tersebut dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector*. Walaupun begitu, tenyata untuk masalah domain yang bersifat nonlinier tidak bisa dipecahkan dengan SVM standar, sehingga perlu dimodifikasi dengan memasukkan fungsi kernel didalamnya. Ide utama dari metode kernel ini adalah memodifikasi SVM dengan

memetakan data *input* nonlinier tersebut ke ruang vektor yang berdimensi lebih tinggi. Sehingga, pada ruang vektor yang baru, *hyperplane* dapat dikonstruksikan. Penjelasan lebih lanjut tentang kernel ini akan disajikan pada Subbab 2.4.

Pada kenyataannya menentukan fungsi kernel yang tepat untuk menyelesaikan klasifikasi yang dikerjakan dengan baik adalah hal yang sulit. Oleh karena itu, seiring dengan berkembangnya penelitian, para peneliti di bidang pembelajaran mesin mencoba untuk mengembangkan pembelajaran dengan kernel yang lebih fleksibel. Beberapa penelitian terbaru menunjukkan bahwa penggunaan banyak kernel (Multi Kernel) memiliki performa yang lebih baik daripada penggunaan satu fungsi kernel (Rakotomamonjy, 2008).

Selain permasalahan pemilihan kernel yang baik selama proses pembelajaran, skema pembelajaran juga menjadi masalah yang cukup penting, seperti diketahui selama ini proses pembelajaran pada algoritma pembelajaran mesin menggunakan bentuk satuan kelompok, artinya semua data pembelajaran memiliki prioritas yang sama selama proses *training*, jika pada suatu kasus terjadi penambahan data pembelajaran baru maka akan terjadi perubahan batas parameter pada data yang telah dilakukan pembelajaran sebelumnya. Ini artinya walaupun penambahan data yang terjadi relatif sedikit dan berpengaruh kecil pada hasil pengklasifikasian namun proses pembelajaran harus diulang dari awal, sehingga seolah-olah pembelajaran yang lalu tidaklah berarti, hal ini tentunya tidaklah efektif terkait waktu dan memori yang digunakan. Untuk itu Gosselin dkk (2010) menggunakan pendekatan yang biasa disebut dengan pembelajaran yang bertambah (*incremental learning)* untuk membangun sebuah sistem temu kembali citra (*image retrieval*) yang lebih dinamis dibandingkan dengan temu kembali citra pada umumnya. Ide dasar dari pendekatan ini adalah dengan memilih data tertentu yang disebut dengan kandidat *support vector* sebagai data pembelajaran baru, kandidat *support vector* adalah data yang memenuhi suatu fungsi evaluasi dimana data tersebut merupakan data yang dekat dengan batas *margin* dari antar kelas atau tidak. Sehingga dengan ini proses pembelajaran tidak harus diulang dari awal.

Berdasarkan latar belakang di atas, pada penelitian ini dibangun model klasifikasi dengan SMV yang berbasis multi kernel dengan pembelajaran yang bertambah dan diterapkan pada aplikasi temu kembali citra (*image retrieval).*

Image retrieval merupakan aplikasi dengan memanfaatkan teknik tertentu untuk mencari atau menemukan citra-citra yang memiliki kemiripan karakter dari citra acuan (*input*). Banyak kemungkinan citra hasil temu kembali adalah citra yang jauh berbeda dengan citra acuan, hal ini menyebabkan tingkat keefektifan dari hasil temu kembali citra menjadi kurang baik. Oleh karena itu, diperlukan model klasifikasi yang baik untuk membangun aplikasi temu kembali citra yang baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan sebelumnya, maka permasalahan yang dibahas dalam penelitian ini adalah sebagai berikut:

a. Bagaimana membangun klasifikasi dengan metode SVM yang berbasis multi kernel dengan pembelajaran yang bertambah serta penerapannya pada temu kembali citra.
b. Bagaimana perbandingan tingkat performa dari temu kembali citra yang dibangun menggunakan metode klasifikasi yang dikerjakan dengan metode SVM standar.

1.3 Batasan Masalah

Batasan masalah dari penelitian ini adalah data citra yang digunakan adalah citra objek tunggal, yaitu citra yang terdiri atas satu objek dominan.

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

a. Membangun klasifikasi dengan metode SVM yang berbasis multi kernel dengan pembelajaran yang bertambah serta penerapannya pada temu kembali citra.
b. Membandingkan tingkat performa dari temu kembali citra yang dibangun menggunakan metode klasifikasi yang dikerjakan dengan metode SVM standar.

1.5 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah memperoleh model klasifikasi dari metode SVM berbasis multi kernel dengan pembelajaran yang bertambah sehingga nantinya dapat diaplikasikan pada aplikasi temu kembali citra atau aplikasi sejenis yang dibangun dengan memanfaatkan algoritma pengklasifikasian. Serta sebagai suatu bentuk kontribusi dalam pengembangan ilmu aplikasi di bidang pembelajaran mesin.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini dijelaskan beberapa pustaka dan dasar teori yang dikaji untuk digunakan dalam pembahasan selanjutnya. Diantaranya adalah penelitian yang relevan yang telah dilakukan sebelumnya serta dijadikan dasar untuk melakukan penelitian, pengertian dan teori tentang pembelajaran mesin, *support vector machine* (SVM), metode kernel, muti kernel, temu kembali citra (*image retrieval*) serta pembelajaran yang bertambah.

2.1 Penelitian yang Relevan

Penelitian tentang pengenalan pola maupun klasifikasi suatu objek telah banyak dilakukan terlebih dalam lingkup jaringan syaraf tiruan, namun pembelajaran dengan kernel bisa dikatakan sebagai hal baru dalam dunia pembelajaran mesin. Penelitian sebelumnya yang dijadikan dasar untuk melakukan penelitian ini adalah penelitian yang dilakukan oleh Gosselin dkk (2010) dalam penelitiannya yang berjudul "*Incremental Kernel Learning for Active Image Retrieval without Global Dictionaries*" dan Zhang dkk (2009) dalam penelitiannya yang berjudul "*A new Incremental Learning Support Vector Machine*". Mereka membangun sebuah metode klasifikasi berdasarkan metode kernel dengan teknik pembelajaran yang bertambah (*incremental*). Artinya, apabila terjadi penambahan data atau informasi baru tidak harus menghapus pengetahuan yang lalu dan mengulang pembelajarannya dari awal dengan data hasil penggabungan antara data lama dan data baru. Serta penelitian yang dilakukan oleh Rakotomamonjy dkk (2008) tentang bagaimana membangun sebuah metode klasifikasi yang berbasis multi kernel serta memperkenalkan algoritma pengklasifikasian berbasis multi kernel yang lebih baik dari metode sebelumnya yang pernah ada yang disebut dengan algoritma *SimpleMKL*.

Gosselin dkk (2010) menggunakan metode kernel dengan pembelajaran yang bertambah dan mengaplikasikannya pada aplikasi temu kembali citra (*image retrieval*). Hasilnya, rata–rata akurasi yang diperoleh adalah 46%, namun dengan

waktu komputasi yang lebih cepat dari metode-metode pembelajaran standar, begitu pula dengan yang dikerjakan oleh Zhang dkk (2009), dimana melalui penelitiannya mereka menunjukkan bahwa klasifikasi yang dilakukan menggunakan SVM dengan pembelajaran yang bertambah membutuhkan waktu komputasi yang lebih cepat daripada SVM standar pada umumnya, namun tetap memiliki akurasi yang baik. Sedangkan Rakotomamonjy dkk (2008) menunjukkan bahwa metode klasifikasi dengan pembelajaran multi kernel tidak hanya mampu diaplikasikan untuk pengklasifikasian biner, regresi, dan *clustering* pada satu kelas maupun banyak kelas, namun juga menunjukkan bahwa metode yang diajukan tersebut memberikan hasil akurasi yang baik.

2.2 Pembelajaran Mesin

Pembelajaran mesin adalah satu disiplin ilmu dari *computer science* yang mempelajari bagaimana membangun komputer/mesin yang cerdas. Pembelajaran mesin dapat didefinisikan sebagai metode komputasional yang menggunakan pengalamannya untuk memperbaiki performa atau membuat prediksi yang akurat, pengalaman disini merujuk pada informasi masa lalu yang didapat dari perekaman data atau proses pengenalan yang dihasilkan dari analisa data (Mohri dkk, 2012).

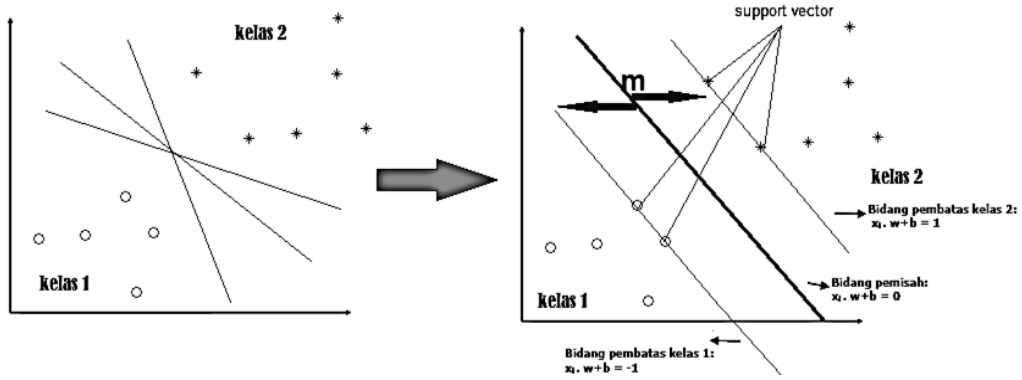
Pada umumnya pembelajaran mesin dibagi menjadi dua tipe. Diantaranya, untuk prediksi yaitu memperkirakan suatu nilai di masa depan berdasarkan informasi yang telah ada sebelumnya atau untuk klasifikasi yaitu menyusun data secara sistematis menurut aturan-aturan yang telah ditetapkan sebelumnya. Tujuan dasar dari pembelajaran mesin adalah untuk mempelajari pemetaaan dari *input* x menjadi *output* y, misalkan diberikan sebuah himpunan *input* dan *output*, $D = \{(x_i, y_i)\}_{i=1}^{N}$, D inilah yang disebut dengan *training set* dan N adalah jumlah dari data *training*. Pada tahap *testing*, sederhananya setiap *training* x_i adalah vektor D-dimensi, yang merepresentasikan *feature*, *attribute*, dan kovariasi. Secara umum x_i bisa berupa objek terstruktur yang kompleks. Misalnya citra, huruf dan kalimat, data *time series,* dll (Murphy dkk, 2012). Pembelajaran mesin sendiri banyak digunakan dalam beberapa aplikasi diantaranya mesin kontrol dan robot, *game*, biologi komputasional, diagnosa medis dll. Hal ini dikarenakan kemampuan dari

pembelajaran mesin yang sangat baik dalam memecahkan permasalahan seperti klasifikasi, regresi, *ranking*, *clustering*, dll (Mohri dkk, 2012).

2.3 *Support Vector Machine* (SVM)

Support vector machine merupakan teknik yang mampu menyelesaikan masalah klasifikasi dengan baik (Rakotomamonjy dkk, 2008; Gosselin dkk, 2010; Athoillah dkk, 2015; Campbell & Ying, 2011; Tomasouw dkk, 2012). SVM sendiri dapat dikategorikan dalam keluarga ANN (*artificial neural network*) atau jaringan syaraf tiruan dalam hal fungsi dan kondisi permasalahan yang dapat diselesaikan, bahkan dapat dikatakan sebagai salah satu yang terbaik dalam hal pengklasifikasian (Mohri dkk, 2012). SVM merupakan metode klasifikasi yang ditemukan oleh Vapnik pada tahun 1998, ide dasar dari metode ini adalah mendefinisikan batas antara dua kelas dengan jarak maksimal berdasarkan data yang terdekat (Clarke dkk, 2009). Berbeda dengan strategi *neural network* yang berusaha mencari *hyperplane* (garis pemisah) antar kelas, SVM berusaha menemukan *hyperplane* yang terbaik pada *input space*. *Hyperplane* pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tersebut dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector* (Campbell dkk, 2011).

SVM merupakan *linear classiffier*, pada umumnya data yang digunakan adalah data yang *linearly separable* yaitu data yang dapat dipisah secara linier. Misalkan terdapat $\{x_1, \dots, x_n\}$ adalah *dataset* dan $y_i \in \{+1, -1\}$ adalah label kelas dari data x_i. Pada Gambar 2.1 berikut dapat dilihat berbagai alternatif bidang pemisah yang dapat memisahkan semua *dataset* sesuai dengan kelasnya. Namun, bidang pemisah terbaik tidak hanya dapat memisahkan data tetapi juga memiliki *margin* yang besar. Data yang berada pada bidang pembatas inilah yang disebut dengan *support vector*.



Gambar 2.1. Bidang pemisah pada SVM

Pada Gambar 2.1 dua kelas dapat dipisahkan oleh sepasang bidang pembatas yang sejajar.

$$x_i.w + b \geq +1 \text{ untuk } y_i = +1$$
$$x_i.w + b \geq -1 \text{ untuk } y_i = -1 \qquad (2.1)$$

w adalah normal bidang (vektor bobot yang tegak lurus terhadap *hyperplane*) dan b adalah bias (posisi bidang relatif terhadap pusat koordinat) (Abe, 2010). Nilai *margin* antara bidang pembatas adalah $\frac{w}{\|w\|}(x_1 - x_2) = \frac{2}{\|w\|}$. Nilai *margin* ini dimaksimalkan dengan tetap memenuhi Persamaan (2.1). Dengan mengalikan b dan w dengan sebuah konstanta, maka dihasilkan nilai *margin* yang dikalikan dengan konstanta yang sama. Oleh karena itu, kendala pada Persamaan (2.1) disebut sebagai *scaling constraint* yang dapat dipenuhi dengan *rescaling* b dan w. Selain itu, karena memaksimalkan $\frac{1}{\|w\|}$ sama saja dengan meminimumkan $\|w\|^2$ dan jika kedua bidang pembatas pada Persamaan (2.1) direpresentasikan dengan Pertidaksamaan (2.2) berikut:

$$y_i(x_i.w + b) - 1 \geq 0 \qquad (2.2)$$

Maka pencarian bidang pemisah terbaik dengan nilai *margin* terbesar dapat dirumuskan sebagai berikut:

$$\min \frac{1}{2}\|w\|^2$$

dengan syarat $y_i(x_i.w + b) - 1 \geq 0$

Permasalahan ini dapat diselesaikan dengan mengubahnya ke formula *lagrangian* dengan menggunakan *lagrange multiplier*. Sehingga permasalahan optimasi menjadi:

$$\min L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i . w + b) + \sum_{i=1}^{n} \alpha_i \qquad (2.3)$$

dengan tambahan kendala, $\alpha_i \geq 0$ (nilai dari koefisien *lagrange*). Dengan meminimalkan L_p terhadap w dan b, maka dari $\frac{\partial}{\partial b} L_p(w, b, a) = 0$ diperoleh

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

dan dari $\frac{\partial}{\partial w} L_p(w, b, \alpha) = 0$,

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad (2.4)$$

Vektor w seringkali bernilai besar (mungkin tak terhingga), tapi nilai α_i terhingga, oleh sebab itu, formula *lagrangian* L_p (*primal problem*) diubah kedalam *dual problem* L_D. Dengan mensubsitusikan Persamaan (2.4) ke L_p diperoleh *dual problem* L_D yaitu:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i \, y_j x_i x_j$$

Sehingga, permasalahan pencarian bidang pemisah terbaik dapat dirumuskan sebagai berikut:

$$\max_{\alpha} \; L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i \, y_j x_i x_j$$

dengan syarat

$$\sum_{i=1}^{n} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0$$

Dengan demikian, dapat diperoleh nilai α_i optimal yang nantinya digunakan untuk menemukan w. Terdapat nilai α_i untuk setiap data *training*. Data *training* yang memiliki nilai $\alpha_i > 0$ adalah *support vector* sedangkan sisanya memiliki nilai

$\alpha_i = 0$. Dengan demikian fungsi keputusan yang dihasilkan hanya dipengaruhi oleh *support vector*. Formula pencarian bidang pemisah terbaik ini adalah permasalahan *quadratic programming*, sehingga nilai maksimum global dari α_i selalu dapat ditemukan. Setelah solusi permasalahan *quadratic programming* ditemukan (nilai α_i), maka kelas dari data pengujian x dapat ditentukan berdasarkan nilai dari fungsi keputusan:

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i x_i x_d + b \qquad (2.5)$$

x_i adalah *support vector*, ns = jumlah *support vector* dan x_d adalah data yang diklasifikasikan. Sedangkan untuk mengklasifikasikan data nonlinier dilakukan dengan menambahkan variable *slack* $\xi_i \geq 0$ dan C sebagai batasan antara memaksimalkan *margin* dan mengurangi jumlah yang salah ketika diklasifikasi pada Pertidaksamaan (2.2) (Steinwart dkk, 2008), sehingga fungsi tujuan dan kendala menjadi:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \qquad (2.6)$$

dengan kendala

$$y_i(x_i.w + b) \geq 1 - \xi_i \quad \forall\, i$$
$$\xi_i \geq 0 \quad \forall\, i$$

Sehingga bentuk primal pada Persamaan (2.3) berubah menjadi:

$$\min L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i\{y_i(x_i.w + b) - 1 + \xi_i\} + \sum_{i=1}^{n} q_i\,\xi_i$$

Seperti pada proses sebelumnya, untuk meminimumkan $L(w, x, \xi_i)$ digunakan $\frac{\partial}{\partial b}L_p(w, x, \xi_i) = 0$, $\frac{\partial}{\partial w}L_p(w, x, \xi_i) = 0$ dan $\frac{\partial}{\partial \xi_i}L_p(w, x, \xi_i) = 0$ maka didapat

$$\frac{\partial}{\partial b}L_p = \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial}{\partial w}L_p = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$\frac{\partial}{\partial \xi_i}L_p = C - \alpha_i - q_i = 0$$

akibatnya,

$$C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}q_i\xi_i = 0$$

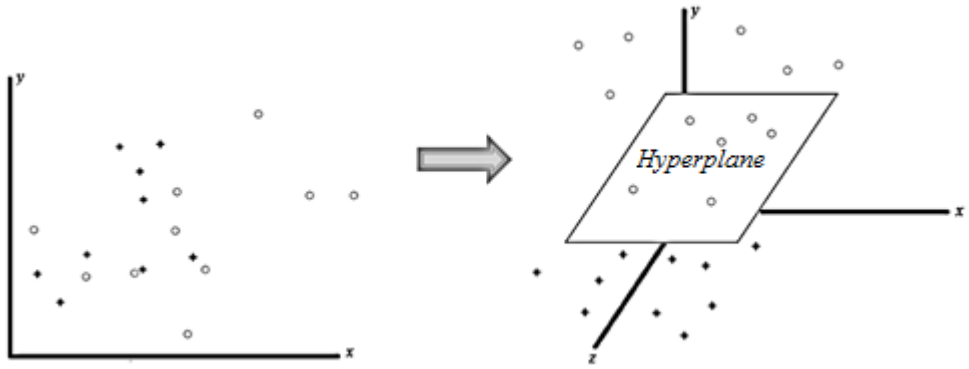
Jadi bentuk *dual* dari permasalahan pencarian bidang pemisah terbaik dapat dirumuskan sebagai:

$$\max\ L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i y_i\,\alpha_j y_j (x_i . x_j)$$

dengan syarat $0 \le \alpha_i \le C, i = 1,2,\ldots,N$ (Lusiyanti dkk, 2014).

2.4 Metode Kernel

Metode kernel adalah pengembangan dari algoritma SVM yang tujuannya adalah untuk menyelesaikan masalah domain yang bersifat nonlinier yang selama ini tidak bisa dipecahkan dengan SVM standar maupun dengan tambahan variabel *slack*, sehingga perlu dimodifikasi dengan memasukkan fungsi kernel didalamnya. Ide utama dari metode kernel ini adalah memodifikasi SVM dengan memetakan data *x* dari *input space* ke ruang vektor yang berdimensi lebih tinggi (*feature space*) dengan fungsi Φ sehingga $\Phi : x \rightarrow \Phi(x)$. Dengan demikian, maka pada ruang vektor yang baru ini, *hyperplane* dapat dikonstruksikan (Scholkopf & Smola, 2001). Ilustrasi dari konsep ini dapat dilihat pada Gambar (2.2) berikut:



Gambar 2.2. Ilustrasi fungsi Φ yang memetakan data ke ruang vektor yang lebih tinggi

Pada ilustrasi yang ditunjukkan oleh Gambar 2.2 tersebut terlihat bahwa kedua jenis data pada awalnya berada pada *input space* berdimensi dua yang tidak dapat dipisahkan secara linier (gambar sebelah kiri). Kemudian pada Gambar 2.2

(sebelah kanan) menunjukkan fungsi Φ memetakan tiap data pada *input space* tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3), sehingga kedua kelas dapat dipisahkan secara linear oleh sebuah *hyperplane*.

Pemetaan tersebut dilakukan dengan tujuan untuk tetap menjaga topologi data, artinya dua data yang semula berjarak dekat pada *input space* akan tetap berjarak dekat pada *feature space*, begitu pula sebaliknya data yang pada awalnya berjarak jauh akan tetap berjarak jauh. Selanjutnya dalam menemukan titik-titik *support vector* pada proses pembelajaran SVM, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi yaitu $\langle \Phi(x_i).\Phi(x_j) \rangle$ (Scholkopf & Smola, 2001). Karena sulitnya menemukan fungsi transformasi dari Φ, maka perhitungan *dot product* tersebut dapat digantikan dengan fungsi kernel $K(x_{i,}x_j)$ dimana fungsi tersebut mendefinisikan transformasi Φ secara implisit. Inilah yang disebut dengan *"kernel trick"* (Hamel, 2009) dirumuskan dengan:

$$K(x_{i,}x_j) = \langle \Phi(x_i).\Phi(x_j) \rangle \tag{2.7}$$

Trik ini mempermudah dalam menentukan *support vector* selama proses pembelajaran SVM, karena dengan trik ini, untuk mengetahui wujud dari fungsi nonlinear Φ cukup hanya dengan mengetahui fungsi kernel yang dipakai. Berikut beberapa jenis fungsi kernel yang biasa digunakan dalam penelitian menurut Scholkopf dan Smola (2001), yaitu:

a) Kernel Linier:

$$K(x_{i,}x_j) = x_{i,}x_j^T \tag{2.8}$$

b) Kernel Polynomial:

$$K(x_{i,}x_j) = (x_i x_j^T + 1)^{\mathrm{p}} \tag{2.9}$$

c) Kernel RBF :

$$K(x_{i,}x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) \tag{2.10}$$

Selanjutnya adalah hasil klasifikasi dari data x yang semula dirumuskan pada Persamaan (2.5) menjadi Persamaan (2.9) berikut:

$$f(x_d) = \sum_{i=1}^{ns} \alpha_i y_i x_i x_d + b$$

$$= \sum_{i=1, x_i \epsilon SV}^{ns} \alpha_i y_i \langle \Phi(x_i). \Phi(x_d) \rangle + b$$

$$= \sum_{i=1, x_i \epsilon SV}^{ns} \alpha_i y_i K(x_i, x_d) + b \qquad (2.11)$$

SV = Support Vector ialah subset dari *training set* yang terpilih sebagai *support vector*, dengan kata lain data x yang berkorespondensi pada $\alpha_i \geq 0$.

2.5 Multi Kernel

Pada dasarnya, fungsi kernel memiliki dua peran penting dalam masalah klasifikasi, pertama adalah fungsi ini mendefinisikan tingkat kesamaan antara dua *feature* misalnya x dan x', sementara fungsi lainnya adalah mendefinisikan pola pembelajaran yang tepat selama proses pengklasifikasian. Pada kenyataannya menentukan kernel yang terbaik selama proses pembelajaran adalah hal sulit, oleh karena itu, seiring dengan berkembangnya penelitian, para peneliti di bidang pembelajaran mesin mencoba untuk mengembangkan pembelajaran kernel yang lebih fleksibel yaitu dengan pembelajaran multi kernel atau lebih dikenal dengan *Multiple Kernel Learning (MKL)* (Rakotomamonjy dkk, 2008). Beberapa penelitian terbaru menunjukkan bahwa pembelajaran multi kernel mampu memberikan hasil yang baik dalam hal memprediksi maupun mengklasifikasi (Shrivastava dkk, 2015; Li dkk, 2014; Liu dkk, 2015; Yeh dkk, 2013)

Menurut Gonen (2011) ada dua pendekatan yang dapat dilakukan dalam proses pembelajaran dengan multi kernel yaitu :

a) Metode satu langkah : adalah metode dengan menghitung kombinasi dari fungsi parameter yang ada dan parameter dasar dari kombinasi *learner* dalam satu fase. Metode ini dapat dilakukan dengan pendekatan berurutan atau pendekatan simultan.
b) Metode dua langkah : adalah metode dengan menggunakan pendekatan berulang pada setiap iterasi. Pertama, *update* fungsi kombinasi parameternya

seraya membenahi parameter dasar *learner*, kemudian *update* parameter dasar *learner* tersebut seraya membenahi fungsi kombinasinya dan berulang seterusnya.

2.5.1 SVM dengan Multi Kernel

Jika pada Persamaan (2.11) fungsi kernel didefinisikan dengan $K(x, x')$ maka pada pembelajaran multi kernel, fungsi tersebut dapat digantikan oleh fungsi kombinasi dari beberapa fungsi kernel dengan $K_1(x, x')$, $K_2(x, x'), \dots, K_M(x, x')$ yang dirumuskan sebagai berikut:

$$K(x, x') = \sum_{m=1}^{M} d_m K_m(x, x') \qquad (2.12)$$

dengan kendala

$$d_m \geq 0, \sum_{m=1}^{M} d_m = 1$$

M adalah jumlah dari fungsi kernel (Rakotomamonjy dkk, 2007). Kernel $K_m(x, x')$ bisa saja merupakan kombinasi dari berbagai fungsi kernel yang berbeda atau satu jenis fungsi kernel yang biasa misalnya Polynomial atau Gaussian namun dengan parameter yang berbeda.

Selanjutnya, jika pada fungsi kernel tunggal jarak optimal dari *hyperplane* didapatkan dengan menyelesaikan Persamaan (2.6) sebagai berikut:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

dengan kendala

$$y_i(x_i.w + b) \geq 1 - \xi_i \quad \forall\, i$$
$$\xi_i \geq 0 \quad \forall\, i$$

maka berdasarkan Rakotomamonjy (2008), pada fungsi multi kernel fungsi keputusan $(x_i.w) + b = f(x) + b$ menjadi $\sum_{m=1}^{M} f_m(x) + b$ untuk setiap f_m merupakan fungsi RKHS (*Repoducing Kernel Hilbert Space*) yang berbeda berdasarkan fungsi kernel K_m, sehingga *hyperplane* optimal bisa didapatkan dengan menyelesaikan Persamaan (2.12) berikut:

$$\min \frac{1}{2}\left(\sum_{m=1}^{M} \|f_m\|\right)^2 + C\sum_{i=1}^{n} \xi_i \qquad (2.13)$$

dengan kendala

$$y_i \sum_{m=1}^{M} f_m(x_i) + y_i b \geq 1 - \xi_i, \quad \forall\, i$$

$$\xi_i \geq 0, \quad \forall\, i$$

Kemudian dengan mengetahui $\sum_{m=1}^{M} d_m = 1$ dan $d_m \geq 0$ maka didapat;

$$\left(\sum_{m=1}^{M} \|f_m\|\right)^2 = \left(\sum_{m=1}^{M} \frac{\|f_m\|}{d_m^{1/2}}\, d_m^{1/2}\right)^2$$

$$\leq \left(\sum_{m=1}^{M} \frac{\|f_m\|^2}{d_m}\right)\left(\sum_{m=1}^{M} d_m\right)$$

$$\leq \left(\sum_{m=1}^{M} \frac{\|f_m\|^2}{d_m}\right) \qquad (2.14)$$

dengan mensubstitusikan Pertidaksamaan (2.14) pada Persamaan (2.13), maka didapat:

$$\min \frac{1}{2}\sum_{m=1}^{M} \frac{1}{d_m}\|f_m\|^2 + C\sum_{i=1}^{n} \xi_i \qquad (2.15)$$

dengan kendala

$$y_i \sum_{m=1}^{M} f_m(x_i) + y_i b \geq 1 - \xi_i\,, \quad \forall\, i$$

$$\xi_i \geq 0 \quad \forall\, i$$

$$\sum_{m=1}^{M} d_m = 1,\ d_m \geq 0 \quad \forall\, m$$

Meskipun terlihat bahwa variabel kendala pada Persamaan (2.15) lebih banyak daripada Persamaan (2.13) namun penyelesaian pada Persamaan (2.15) ini lebih efisien. Kemudian untuk menyelesaikan permasalahan tersebut maka Persamaan (2.15) diubah ke dalam bentuk *lagrange* sehingga permasalahan optimasinya menjadi:

$$\min L_p = \frac{1}{2}\sum_{m=1}^{M}\frac{1}{d_m}\|f_m\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$+ \alpha\left\{0 - \sum_{i=1}^{n}\left(y_i\sum_{m=1}^{M} f_m(x_i) + y_i b - 1 + \xi_i\right)\right\} + \beta\left(0 - \sum_{i=1}^{n}\xi_i\right)$$
$$+ \lambda\left\{0 - \left(\sum_{m=1}^{M} d_m - 1\right)\right\} + \mu\left(0 - \sum_{m=1}^{M} d_m\right)$$
$$= \frac{1}{2}\sum_{m=1}^{M}\frac{1}{d_m}\|f_m\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$+ \sum_{i=1}^{n}\alpha_i\left(1 - y_i\sum_{m=1}^{M} f_m(x_i) - y_i b - \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i$$
$$+ \lambda\left(\sum_{m=1}^{M} d_m - 1\right) - \sum_{m=1}^{M}\mu_m d_m \qquad (2.16)$$

dengan α dan β adalah pengali *lagrange* yang sesuai dengan batasan dari fungsi SVM sedangkan λ dan μ merujuk pada batasan dari d_m.

Langkah selanjutnya untuk mendapatkan solusi dari permasalahan tersebut adalah meminimumkan variabel–variabel seperti f_m, b, ξ dan d_m, dengan demikian, maka

a.
$$\frac{\partial L}{\partial f_m} = \frac{1}{d_m} f_m(.) - \sum_{i=1}^{n}\alpha_i y_i K(., x_i)$$
$$\frac{1}{d_m} f_m(.) = \sum_{i=1}^{n}\alpha_i y_i K(., x_i)$$

b.
$$\frac{\partial L}{\partial b} = 0$$
$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

c.
$$\frac{\partial L}{\partial \xi} = 0$$
$$C - \alpha_i - \beta_i = 0$$

$d.$ $$\frac{\partial L}{\partial d_m} = 0$$

$$-\frac{1}{2}\frac{\|f_m\|^2}{{d_m}^2} + \lambda - \mu_m = 0 \qquad (2.17)$$

Kemudian, untuk *dual problem*, dengan menghilangkan d_m sehingga $f_m(.) = 0$ maka didapatkan

$$L_p = \frac{1}{2}\sum_{m=1}^{M}\frac{1}{d_m}\|f_m\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$+\sum_{i=1}^{n}\alpha_i\left(1-y_i\sum_{m=1}^{M}f_m(x_i) - y_i b - \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i$$
$$+\lambda\left(\sum_{m=1}^{M}d_m - 1\right) - \sum_{m=1}^{M}\mu_m d_m$$
$$= C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1- y_i b - \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i - \lambda$$
$$= (C - \alpha_i - \beta_i)\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\, y_i b - \lambda$$

Berdasarkan Persamaan (2.17) a dan b maka didapatkan

$$L_p = 0 + \sum_{i=1}^{n}\alpha_i - 0 - \lambda$$

sehingga *dual problemnya* didapatkan

$$\max \sum_{i=1}^{n}\alpha_i - \lambda \qquad (2.18)$$

dengan kendala

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \quad \forall\, i$$

$$\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \le \lambda, \qquad \forall\, m$$

2.5.2 Penyelesaian SVM dengan Multi Kernel

Berdasarkan penjelasan sebelumnya diketahui bahwa penyelesaian dari klasifikasi SVM dengan multi kernel adalah dengan menyelesaikan persamaan optimasi dari Persamaan (2.6) dimana kemudian dengan ditambahkannya fungsi multi kernel sehingga didapatkan penambahan kendala pada Persamaan (2.12), maka berdasarkan Rakotomamonjy dkk (2007) dapat dituliskan persamaan optimasinya sebagai berikut:

$$\min_{d} J(d) \tag{2.19}$$

dengan syarat

$$d_m \geq 0,$$

$$\sum_{m=1}^{M} d_m = 1$$

dengan $J(d)$ berupa persamaan fungsi tujuan dengan kendala sebagai berikut:

$$\min \frac{1}{2} \sum_{m=1}^{M} \frac{1}{d_m} \|f_m\|^2 + C \sum_{i=1}^{n} \xi_i \tag{2.20}$$

dengan kendala

$$y_i \sum_{m=1}^{M} f_m(x_i) + y_i b \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

kemudian dengan mengacu pada Persamaan (2.18) dan memasukkan nilai λ dan $\sum_{m=1}^{M} d_m = 1$ maka didapatkan *dual problem* dari Persamaan (2.20) sebagai berikut:

$$\max_{\alpha} -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{m}^{M} d_m K_m(x_i, x_j) + \sum_{i=1}^{n} \alpha_i \tag{2.21}$$

dengan

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \ \forall i$$

Persamaan (2.21) adalah masalah *quadratic programming* (QP) dengan pembatas yang linier. Melakukan proses *training* pada SVM sama dengan halnya menyelesaikan *problem convex optimization*. Oleh karena itu, solusi dari SVM adalah unik (dengan asumsi bahwa $K_m(x_i, x_j)$ adalah *positive definite*) dan global optimal (Rakotomamonjy, 2008). $J(d)$ merupakan *objective function* dari Persamaan (2.20). Sehingga, dengan mengacu pada Persamaan (2.21), $J(d)$ juga dapat dituliskan sebagai berikut:

$$J(d) = -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i^*\alpha_j^* y_i y_j \sum_{m}^{M} d_m K_m(x_i, x_j) + \sum_{i=1}^{n}\alpha_i^* \qquad (2.22)$$

dengan α^* adalah solusi optimal dari α, nilai ini dapat diperoleh dengan perhitungan SVM standar dengan berbagai metode pada umumnya, yang nantinya nilai setiap data x dengan $\alpha^* > 0$ merupakan *support vector* yang dapat digunakan untuk proses pembelajaran secara bertambah pada fase selanjutnya.

Untuk memeriksa kondisi optimal yang tepat dapat dilakukan melalui *Karush-Kuhn-Tucker* (KKT) *condition* atau dengan *duality gap* (perbedaan *objective value* antara *primal* dan *dual*) dimana seharusnya bernilai nol jika optimal. Berdasarkan Persamaan (2.18) maka MKL *duality gapnya* didapat:

$$DG = \mathrm{J}(d^*) - \left(\max \sum_{i=1}^{n}\alpha_i - \lambda\right)$$

$$\mathrm{DG} = \mathrm{J}(d^*) - \sum_{i=1}^{n} a_i^* + \frac{1}{2}\max_{m}\sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* K(x_i, x_j) \qquad (2.23)$$

Jika $\mathrm{J}(d^*)$ dihitung berdasarkan Persamaan (2.21), maka MKL Persamaan (2.23) tersebut bisa juga dihitung menggunakan algoritma SVM kernel tunggal (DG_{SVM}) dengan kondisi bahwa hasil dari perhitungan DG_{SVM} tersebut bernilai optimal atau $\mathrm{DG}_{SVM} = 0$. Sehingga DG_{SVM} dapat ditulis sebagai berikut:

$$\mathrm{DG}_{SVM} = \mathrm{J}(d^*) + \frac{1}{2}\max_{m}\sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* \sum_{m}^{M} d_m K_m(x_i, x_j) - \sum_{i=1}^{n} a_i^*$$

$$\mathrm{J}(d^*) = \mathrm{DG}_{SVM} - \frac{1}{2}\max_{m}\sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* \sum_{m}^{M} d_m K_m(x_i, x_j) + \sum_{i=1}^{n} a_i^*$$

Sehingga MKL *duality gap* pada Persamaan (2.23) menjadi:

$$\mathrm{DG} = \left(\mathrm{DG}_{SVM} - \frac{1}{2}\max_{m} \sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* \sum_{m}^{M} d_m K_m(x_i, x_j) + \sum_{i=1}^{n} a_i^* \right)$$
$$- \sum_{i=1}^{n} a_i^* + \frac{1}{2}\max_{m} \sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* K(x_i, x_j)$$
$$= \mathrm{DG}_{SVM} - \frac{1}{2}\max_{m} \sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* \sum_{m}^{M} d_m K_m(x_i, x_j)$$
$$+ \frac{1}{2}\max_{m} \sum_{i,j=1}^{n} a_i^* a_j^* y_i^* y_j^* K(x_i, x_j)$$

Seperti yang telah dijelaskan sebelumya bahwa untuk memeriksa kondisi optimal dapat dilakukan dengan *KKT Condition* atau dengan *Duality Gap*. Berdasarkan Persamaan (2.19) maka turunan pertama yang didapatkan berdasarkan *KKT condition* adalah

$$\frac{\partial J}{\partial d_m} + \lambda - \mu_m = 0, \quad \forall m,$$
$$\mu_m \, . \, d_m = 0, \quad \forall m$$

dengan λ dan μ_m merupakan pengali *lagrange* untuk persamaan dan pertidaksamaan dari kendala Persamaan (2.19). Dari kondisi tersebut maka dapat dituliskan

$$\frac{\partial J}{\partial d_m} + \lambda - \mu_m = 0$$
$$\frac{\partial J}{\partial d_m} + \lambda - \frac{0}{d_m} = 0$$

Sehingga didapatkan:

$$\frac{\partial J}{\partial d_m} = -\lambda \ untuk \ d_m > 0$$
$$\frac{\partial J}{\partial d_m} \geq -\lambda \ untuk \ d_m = 0$$

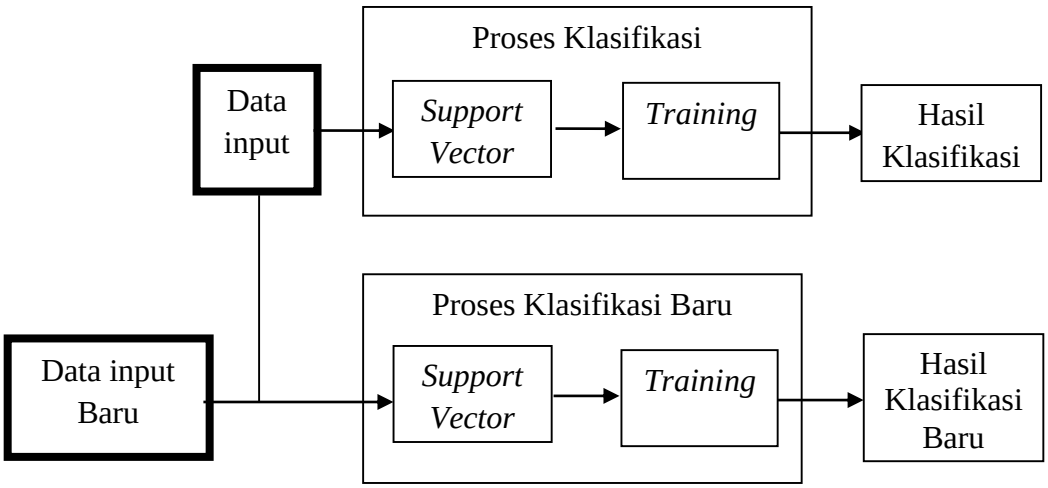
2.6 Pembelajaran yang Bertambah (*Incremental Learning*)

Pada umumnya, pelatihan (*training*) pada algoritma pembelajaran mesin menggunakan bentuk satuan kelompok, artinya semua data *training* akan memiliki prioritas yang sama selama *training*. Sehingga, jika pada suatu kasus terjadi penambahan data pembelajaran baru maka akan terjadi perubahan pada batas parameter pada data yang telah dilakukan pembelajaran sebelumnya. Ini artinya walaupun penambahan data yang terjadi relatif sedikit dan berpengaruh kecil pada hasil pengklasifikasian namun proses pembelajaran harus diulang dari awal, sehingga seolah-olah pembelajaran yang lalu tidaklah berarti, hal ini tentunya tidaklah efektif terkait waktu dan memori yang digunakan.

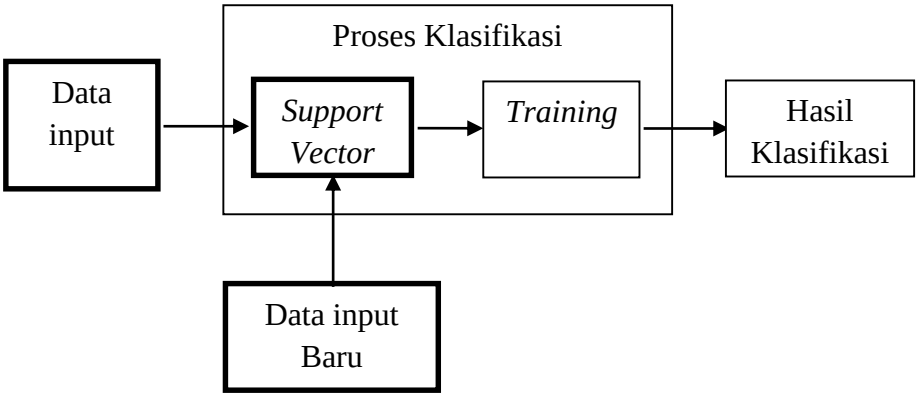
Pada SVM, yang mempengaruhi fungsi keputusan hasil *training* adalah *support vector* (data yang berada pada perbatasan antar kelas). Karakteristik ini dapat menyelesaikan masalah utama pembelajaran mesin yang menjadi sangat lambat ketika dilakukan *training* dengan menggunakan data dalam jumlah besar. Caranya adalah dengan memilih kandidat *support vector* sebelum *training*, sehingga dengan cara ini diharapkan mampu mengurangi jumlah data pada proses *training* dan pada akhirnya mengurangi kebutuhan waktu dan memori (Shinya, 2006).

Jadi dengan kata lain, penggunaan data *support vector* sebagai data *training* pada pembelajaran selanjutnya, membuat model SVM baru yang terbentuk tidak berbeda jauh dengan model SVM sebelumnya yang dilakukan dengan cara standar yaitu dengan menggunakan keseluruhan dari data *training*, hal ini dikarenakan pada setiap pembentukan model SVM dengan cara bertambah (*incremental)*, model SVM tersebut akan mengingat informasi penting batas-batas pemisah kelas dari pembelajaran data yang ada sebelumnya, dan informasi penting inilah yang berperan besar dalam membentuk batas pemisah baru pada SVM yang baru tersebut sehingga model SVM baru yang terbentuk tidak akan menghasilkan sesuatu yang jauh berbeda dengan apa yang dihasilkan oleh SVM sebelumnya (Dominiconi, 2001).

Gambar 2.3 dan Gambar 2.4 berikut ini merupakan ilustrasi bagaimana perbedaan antara sistem dengan pembelajaran standar dengan pembelajaran yang bertambah.



Gambar 2.3. Skema pembelajaran standar



Gambar 2.4. Skema pembelajaran yang bertambah

Jadi terlihat dari Gambar 2.3 dan Gambar 2.4 perbedaan antara proses pembelajaran standar dengan pembelajaran yang bertambah, bahwa pada pembelajaran standar akan selalu melakukan proses *training* yang baru seiring dengan penambahan data atau pembentukan kelas baru, sedangkan pada pembelajaran yang bertambah, cukup dengan memperbarui hasil pembelajaran yang lalu.

Jika model hasil *training* yang lama ingin diperbarui seiring dengan adanya data *training* baru yang didapat, maka dari data *training* yang lama cukup diambil sebagian data yang menjadi *support vector*. *Support vector* adalah data yang tepat berada pada bidang pembatas $f(x) = \pm 1$ (lihat Gambar 2.1). Sedangkan pada

setiap proses *training* dengan SVM, akan didapatkan tiga kondisi yaitu beberapa data x_i yang memenuhi $\alpha_i = 0$ dimana data tersebut merupakan data yang berada diluar dari *margin*, beberapa data x_i yang memenuhi $0 \leq \alpha_i \leq C$ atau disebut data *support vector*, serta data dengan $C = \alpha$ merupakan data yang berada didalam *margin*. Itu artinya untuk menentukan apakah sebuah data merupakan kandidat *support vector* dapat dilakukan dengan cara memeriksa data yang memenuhi:

$$\alpha_i = 0 \rightarrow |f(x_i)| \geq 1 \quad (2.24)$$

$$0 \leq \alpha_i \leq C \rightarrow |f(x_i)| = 1 \quad (2.25)$$

$$\alpha = C \rightarrow |f(x_i)| \leq 1 \quad (2.26)$$

Jika data tidak memenuhi Persamaan (2.25) maka data tersebut merupakan data yang tidak mempengaruhi fungsi keputusan hasil *training*, sehingga pada proses *training* selanjutnya data tersebut akan dibuang.

Seperti telah diketahui sebelumnya dari Persamaan (2.21) bahwa *dual problem* dari SVM adalah:

$$\max_{\alpha} L_D = -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \sum_{m}^{M} d_m K_m(x_i, x_j) + \sum_{i=1}^{n} \alpha_i$$

dengan

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \ \forall i$$

Kemudian jika kita misalkan

$$H_{i,j} = y_i y_j \sum_{m}^{M} d_m K_m(x_i, x_j)$$

Maka

$$\max_{\alpha} -L_D = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i H_{ij} \alpha_j - \sum_{i=1}^{n} \alpha_i$$

Kemudian jika kita turunkan persamaan tersebut terhadap α maka didapat

$$P_i = \frac{\partial L}{\partial \alpha_i} = \frac{1}{2} \sum_{i,j=1}^{n} H_{ij} \alpha_j - 1$$

Menurut Zhang dkk (2009), dengan penambahan data baru s maka P_i berubah mengikuti:

$$\Delta g_i = H_{i,s}\Delta\alpha_s + \sum_{i,j=1}^{n} H_{ij}\alpha_j$$

$$0 = y_s\Delta\alpha_s + \sum_{j\in 1}^{n} y_j\Delta\alpha_j$$

Sehingga, jika pada sebuah sistem ditambahkan data baru s, dimana s merupakan *support vector* maka akan didapatkan :

$$P_s = H_{ss}\Delta\alpha_s + \sum_{s,j}^{n} H_{sj}\left(\alpha_j + \Delta\alpha_j\right) - 1 = 0$$

Sehingga

$$\Delta\alpha_s = \frac{\sum_{j\in s}^{n} H_{sj}\alpha_j - 1}{H_{ss}} \qquad (2.27)$$

dan

$$\alpha = (\alpha + \Delta\alpha, \alpha_s)^T \qquad (2.28)$$

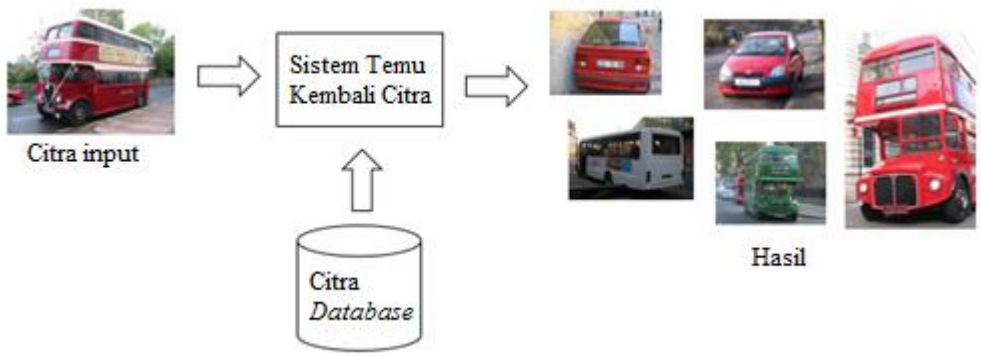
Jadi, jika terdapat penambahan data dimana data tersebut merupakan *support vector* maka α diperbarui menurut Persamaan (2.27) dan (2.28).

2.7 Temu Kembali Citra (*Image Retrieval*)

Temu kembali citra (*Image retrieval)* merupakan aplikasi dengan memanfaatkan teknik tertentu untuk mencari atau menemukan citra-citra yang memiliki kemiripan karakter dari citra acuan (*input*). Dalam hal ini, temu kembali citra dapat diimplementasikan dengan membandingkan fitur-fitur hasil dari ekstraksi citra atau dengan cara yang lain, fitur-fitur dari citra yang diolah bisa didapatkan dengan berbagai cara. Pada umumnya, informasi yang digunakan dalam proses temu citra kembali adalah histogram dari citra, hal ini dikarenakan histogram merepresentasikan karakteristik dari sebuah citra (Lei dkk, 2015). Dalam nilai histogram, bentuk dan konsentrasi dari warna sebuah citra akan selalu sama untuk objek citra yang serupa walaupun objek tersebut berbeda warna satu sama lain. Selain itu, kecepatan proses dan kebutuhan memori yang rendah adalah alasan yang

membuat metode ini lebih banyak digunakan untuk temu kembali citra daripada metode lainnya (Kumar dkk, 2013).

. Temu kembali citra merupakan bagian dari studi *pattern recognition*, jika *pattern recognition* dapat diartikan sebagai bentuk pengenalan sistem atau cara kerja suatu mesin dari *input* sampai pada *output* dalam sembarang objek, maka temu kembali citra lebih berfokus pada pengenalan objek citra, jadi pada dasarnya temu kembali citra ini adalah sebuah sistem yang membantu untuk mengenali sebuah citra dari konten visualnya (Datta, 2008). Lebih lanjut lagi aplikasi temu kembali citra ini dapat dikatakan sebagai sebuah kamus yang membantu penggunanya untuk menemukan citra tertentu dalam sebuah *dataset* yang berisi ribuan bahkan jutaan citra yang berbeda. Gambar 2.5 berikut ini menunjukkan bagaimana secara garis besar cara kerja dari temu kembali citra sehingga dapat dikatakan sebagai kamus dengan objek utama adalah citra.



Gambar 2.5 Skema temu kembali citra

Temu kembali citra (*image retrieval*) merupakan sebuah teknologi yang dapat membantu kemudahan manusia untuk mendapatkan citra serupa atau informasi citra tertentu berdasarkan citra yang menjadi acuan (*input*), dengan bertambah pesatnya teknologi komputasi serta teknologi citra digital, metode untuk temu kembali citra yang efektif menjadi sebuah tantangan tersendiri. Seperti diketahui, pada saat ini aplikasi semacam ini memiliki peran yang penting dalam bidang pengobatan, geografi, keamanan, dan berbagai aplikasi database lainnya. Beberapa aplikasi yang popular saat ini dengan konsep temu citra kembali diantaranya adalah *MIT's Photobook, Google image, VisualSeek, PicSOM dll* (Dinesh, 2014)

BAB 3
METODE PENELITIAN

Pada bab ini dijelaskan metode penelitian tesis yang digunakan untuk menyelesaikan permasalahan pada tesis ini.

3.1 Tahapan Penelitian

Metode penelitian yang dilakukan dalam penelitian ini meliputi tahapan-tahapan sebagai berikut:

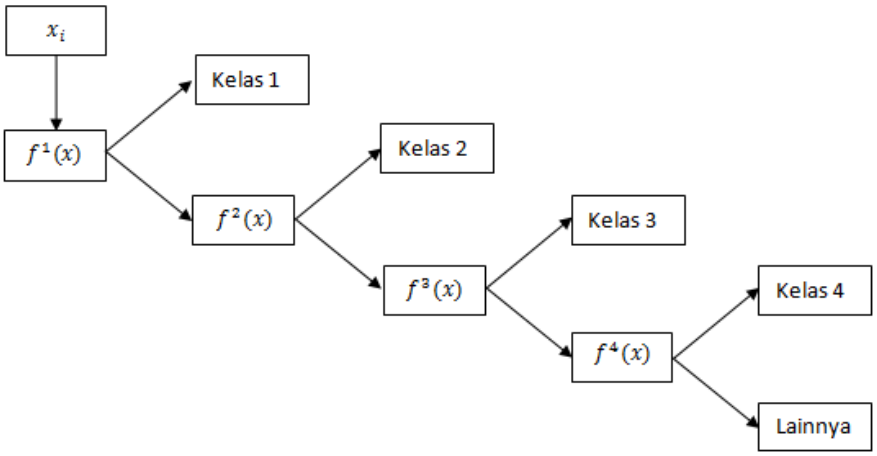
A. Studi Literatur.

Pada tahap ini diawali dengan mempelajari penelitian yang relevan diantaranya yaitu: "*Incremental Kernel Learning for Active Image Retrieval without global dictionaries*" (Gosselin, 2010), "*A new Incremental Learning Support Vector Machine*" (Zhang, 2009) dan "*SimpleMKL*" (Rakotomamonjy, 2008), serta mempelajari teori-teori SVM, pembelajaran dengan kernel, multi kernel, dan pembelajaran yang bertambah dari beberapa pustaka yang ada.

B. Membangun klasifikasi menggunakan metode SVM berbasis multi kernel.

Pada tahap ini, langkah pertama yang dilakukan adalah membangun klasifikasi dengan SVM yang selanjutnya dimodifikasi dengan memasukkan fungsi kernel dimana kernel tersebut didapatkan dari hasil kombinasi dari beberapa fungsi kernel yang telah ada sebelumnya, sehingga menjadi klasifikasi SVM dengan multi kernel.

SVM pada dasarnya adalah pengklasifikasi biner (hanya dua kelas), maka untuk membentuk klasifikasi dengan banyak kelas diperlukan sebuah pendekatan lain dalam pembelajarannya yaitu pendekatan *One-against-all*. Pada metode ini setiap model klasifikasi ke-*i* di-*training* menggunakan keseluruhan data, misalnya akan dibentuk klasifikasi dengan 5 kelas maka skema pengklasifikasiannya adalah sebagai berikut:



Gambar 3.1. Ilustrasi klasifikasi *one-against-all*

Gambar 3.1 menunjukkan bagaimana melakukan proses *training* SVM *Multiclass* (berjumlah 5 kelas) dengan pendekatan *One-against-all*. Langkah pertama adalah dengan mengklasifikasikan data *input* x_i menjadi 2 kelas yaitu kelas 1 dengan 4 kelas sisanya sehingga menghasilkan model klasifikasi yang disimbolkan dengan $f^1(x)$, kemudian langkah selanjutnya adalah mengklasifikasikan keempat kelas yang tersisa dengan model klasifikasi $f^2(x)$ menjadi 2 kelas lagi yaitu kelas 2 dengan 3 kelas sisanya, begitu seterusnya sehingga terbentuk klasifikasi yang terdiri dari 5 kelas.

Selanjutnya, dari model SVM Multi Kernel tersebut akan dimodifikasi sehingga didapatkan model klasifikasi dengan pembelajaran yang bertambah. Seperti yang telah disebutkan pada bab sebelumnya bahwa pembelajaran yang bertambah dilakukan dengan mendapatkan *support vector* dari hasil pembelajaran sistem sebelumnya yang selanjutnya data tersebut digunakan sebagai data *training* pada pembelajaran berikutnya bersama dengan data baru yang ditambahkan pada sistem. Sehingga pada fase *update* data ini terjadi proses *merging* data antara data lama dan data baru. Misalkan X merupakan data awal dan Y adalah data baru, sedangkan data baru hasil penggabungan data X dan Y adalah Z, dapat dituliskan sebagai berikut:

$$Z = X \cup Y$$
$$= \{X_1, X_2, X_3, \dots, X_N, Y_1, Y_2, Y_3, \dots, Y_M\} \in \mathbb{R}^N$$

dengan X_N data pada kelompok data ke-t dan Y_M data pada kelompok data ke-(t+1).

Jadi, misalkan *dataset* lama adalah x_a dan *dataset* baru adalah x_b, maka langkah untuk proses pembelajaran bertambah pada sistem yang dibangun adalah sebagai berikut :

1. Lakukan *training* SVM dengan data x_a
2. Dapatkan *support vector*, x_a yang memenuhi $f(x) = \pm 1$
3. Tambahkan data x_b sehingga data $x_a = x_a \cup x_b$
4. Ulangi langkah 1.

C. Penerapan model klasifikasi yang telah dibangun pada temu kembali citra.

Secara umum proses membangun aplikasi temu kembali citra dibagi dalam beberapa tahap yaitu:

1. Ekstraksi fitur : Ekstraksi fitur dilakukan dengan mencari histogram dari setiap citra yang menjadi objek dari *input* pencarian atau citra dari *database* yang kemudian diklasifikasikan menurut kategorinya masing-masing. Penggunaan histogram citra ini dikarenakan fitur dan warna setiap citra yang membedakan antara citra satu dengan yang lainnya direpresentasikan oleh histogram (Kumar dkk, 2013; Lei dkk, 2015).

   Histogram dari sebuah citra didapatkan dengan menghitung frekuensi kemunculan nisbi (*relative*) dari intensitas pada citra tersebut. Misalkan sebuah citra memiliki *L* derajat keabuan (0 sampai L-1). Secara matematis histogram citra dihitung dengan rumus:

   $$h_i = \frac{n_i}{n}, \quad i = 0,1, \dots, L-1 \qquad (3.1)$$

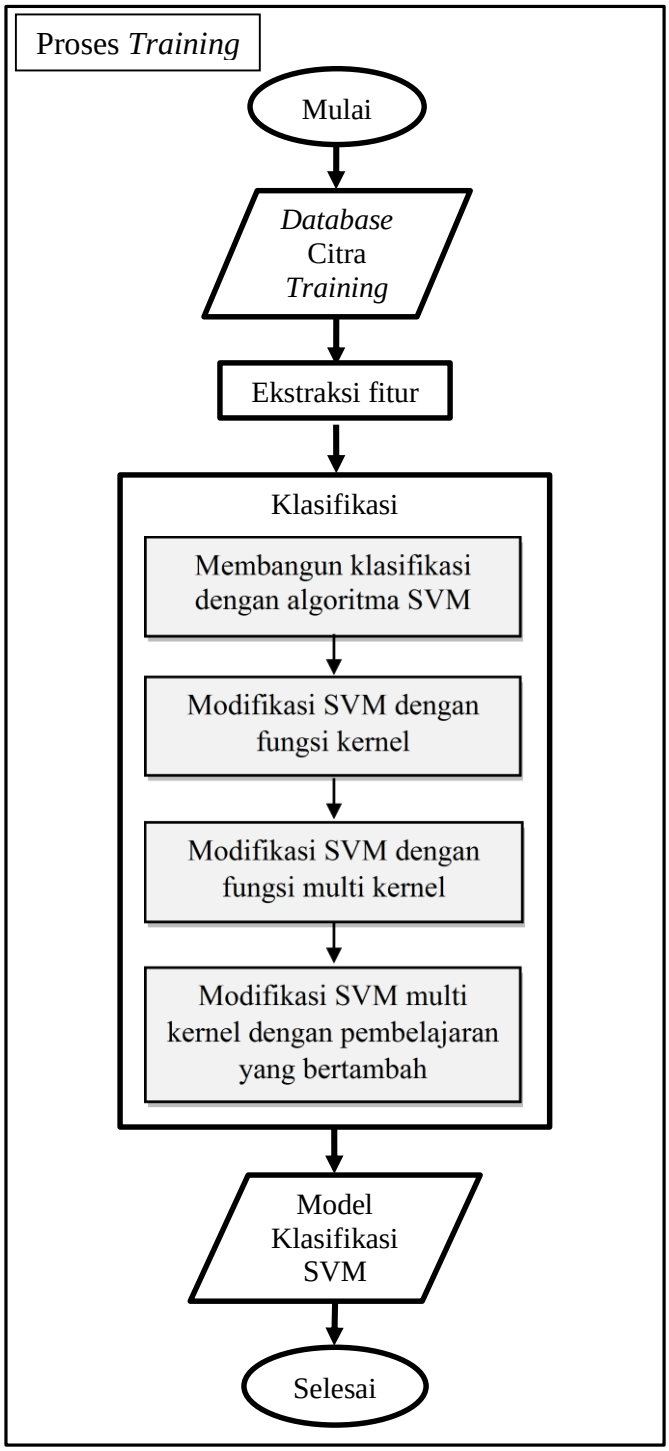
   dengan :

   n_i adalah jumlah *pixel* yang memiliki derajat keabuan i

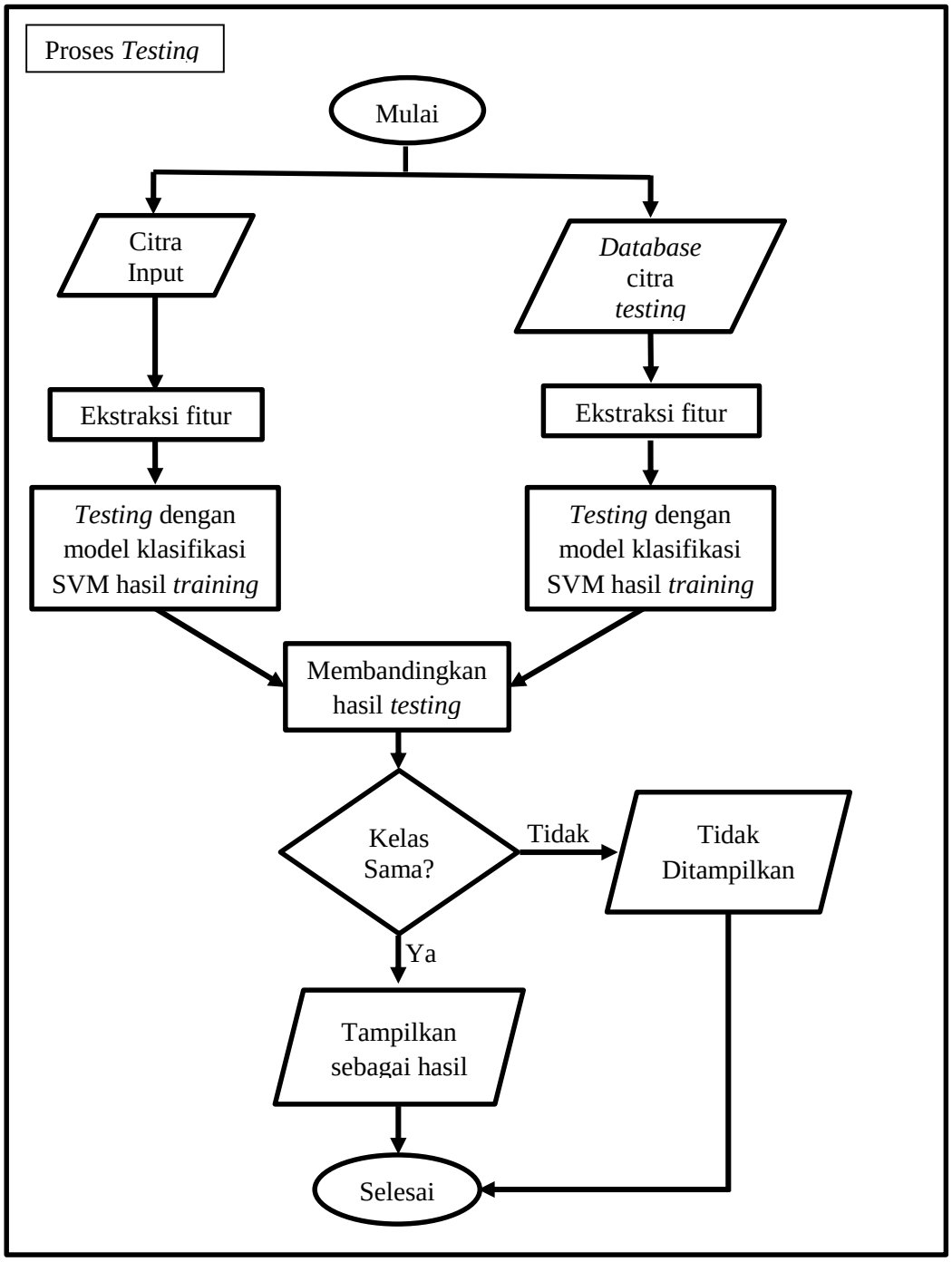
   n adalah jumlah seluruh *pixel* di dalam citra
2. Klasifikasi Citra (*Training*) : Setelah histogram didapatkan, histogram dari masing-masing citra tersebut diklasifikasikan dengan metode klasifikasi yang telah dibentuk sebelumnya.
3. Tahap *retrieval* (*Testing*) : pada tahap ini citra *input* dan citra *database* yang telah diklasifikasikan saling dibandingkan, jika hasil dari perbandingan tersebut menunjukkan kelas yang sama, maka citra tersebut akan

ditampilkan sebagai hasil pencarian dari citra yang serupa dengan citra yang menjadi acuan (*input*).

Gambar 3.2 dan 3.3 berikut ini adalah diagram alir dari proses implementasi dari temu kembali citra yang dibangun:



Gambar 3.2. Diagram alir proses *training* temu kembali citra



Gambar 3.3. Diagram alir proses *testing* temu kembali citra

D. Analisis hasil dari aplikasi yang telah dibangun.

Pada tahap ini, dibahas performa dari sistem yang telah dibangun serta analisa dari hasil yang didapatkan. Kemudian juga dilakukan pembahasan perbandingan performa dari sistem yang telah dibuat dengan metode yang diajukan, dengan sistem yang dibuat dengan metode lainnya.

Untuk mengukur performa dari sistem yang dibangun (aplikasi temu kembali citra) maka dilakukan perhitungan *precision, recall* dan akurasinya serta waktu komputasi selama proses *training* maupun proses *testing* (pencarian citra) dengan satuan detik. Perhitungan dengan ketiga kriteria *precision, recall* dan akurasi dilakukan karena ketiga kriteria dapat merepresentasikan performa dari sistem temu kembali citra dengan benar. Sebab jika hanya menggunakan salah satu kriteria *preicision, recall* atau akurasi saja dapat menimbulkan bias yang sangat fatal. Hal ini dikarenakan penilaian dalam pencarian informasi khususnya dengan *query* berupa citra seperti temu citra kembali bersifat obyektif. Oleh karena itu dibutuhkan penilaian dari *precision* dan *recall* dimana kedua nilai tersebut berketerbalikan satu sama lain. (Datta dkk, 2008).

Precision merupakan perhitungan dari tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi, sedangkan akurasi sendiri didefinisikan dengan tingkat kedekatan antara nilai prediksi dengan nilai sebenarnya. Tabel 3.1 berikut menggambarkan hubungan hasil dari nilai prediksi sistem dengan nilai data sebenarnya yang disebut dengan *confussion matrix* (Abe, 2010):

Tabel 3.1 *Confusion matrix* nilai prediksi dan sebenarnya

		Nilai Prediksi	
		True	***False***
Nilai Sebenarnya	***True***	*True Positive*	*False Negative*
	False	*False Positive*	*True Negative*

- *True Positive* dapat didefinisikan sebagai jumlah data positif (benar) yang dapat diklasifikasikan oleh sistem dengan benar.
- *False Negative* merupakan jumlah data positif yang salah diklasifikasikan
- *False Positive* adalah jumlah data negatif yang salah diklasifikasikan
- *True Negative* merupakan jumlah data negatif yang diklasifikasikan dengan benar

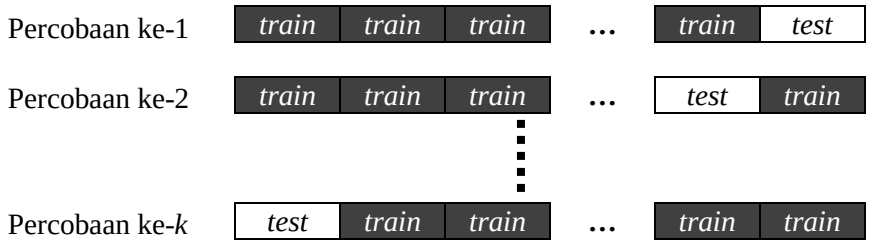
Sedangkan untuk mengukur nilai dari *Precision, Recall* dan Akurasi dapat dihitung dengan rumus sebagai berikut:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \times 100\% \quad (3.2)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \times 100\% \quad (3.3)$$

$$\text{Akurasi} = \frac{True\ Positive + True\ Negative}{\text{Jumlah keseluruhan data}} \times 100\% \quad (3.4)$$

Untuk validasi dari metode yang telah dibentuk, digunakan metode *k-fold cross-validation* (Mohri, 2012), dimana data dibagi menjadi kelompok-kelompok berjumlah k, dengan pembagian data x berjumlah $(k-1)$ kelompok untuk *training* dan y berjumlah 1 kelompok untuk *testing*. Kemudian proses tersebut diulang sebanyak k kali dengan data *training* dan *testing* yang selalu berbeda dengan skema untuk percobaan sebagai berikut: misalkan setiap kelompok data disimbolkan dengan K, maka percobaan yang pertama data $x = (K_1, K_2, \dots, K_{k-1})$ sedangkan data $y = K_k$ kemudian percobaan kedua data $x = (K_1, K_2 \dots, K_{k-2}, K_k)$ sedangkan data $y = K_{k-1}$ dan begitu seterusnya sampai semua data berotasi sepenuhnya, Gambar 3.4 berikut adalah ilustrasi dari *k-fold cross validation.*



Gambar 3.4. Skema *k-fold cross validation*

E. Penyusunan *paper*

Pada tahap ini dilakukan penulisan *paper* hasil penelitian yang dilakukan untuk keperluan publikasi dari penelitian yang telah dilakukan.

F. Publikasi *Paper*

Mempublikasikan hasil penelitian yang telah disusun dalam bentuk *paper* yang telah dilakukan sebelumnya.

G. Penyusunan hasil penelitian

Pada tahap ini dilakukan penulisan laporan hasil penelitian yang dilakukan mulai dari tahap studi literatur sampai dengan analisis hasil dari aplikasi yang telah dibangun hingga mendapat kesimpulan seperti yang diinginkan.

BAB 4
PEMBAHASAN

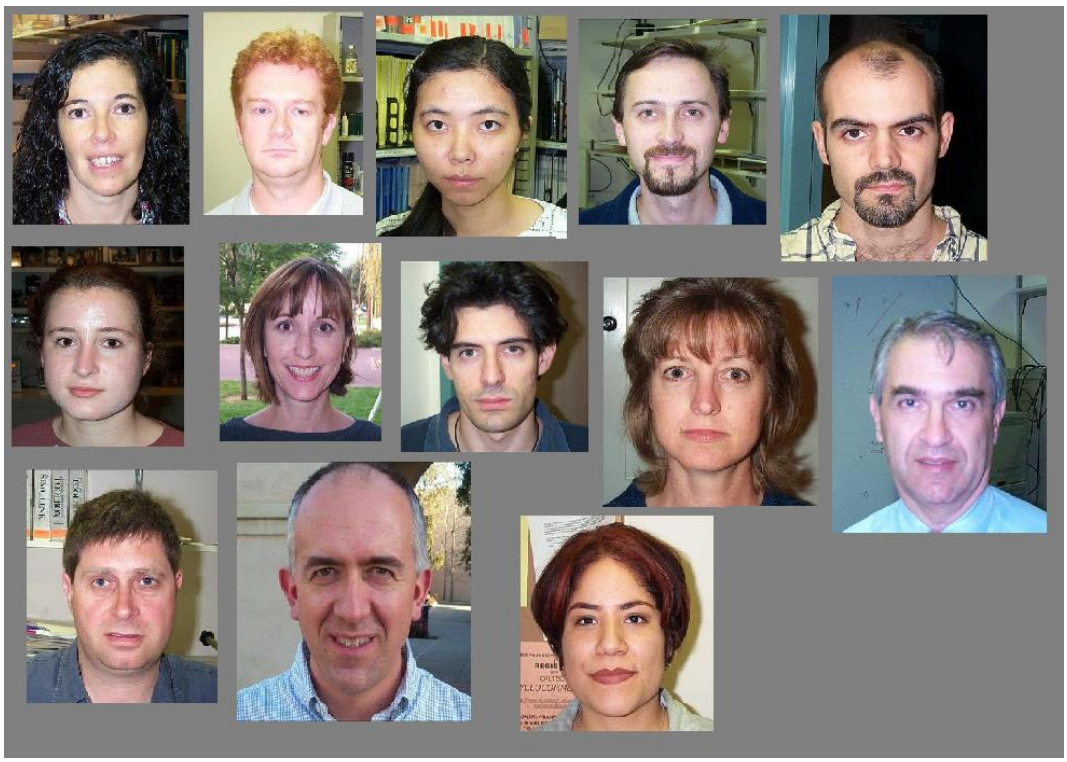
Bab ini dibahas mengenai bagaimana membentuk metode klasifikasi dengan SVM Multi kernel dengan pembelajaran yang bertambah dan penerapannya pada temu kembali citra serta bagaimana hasil sistem yang dibangun dan perbandingannya dengan SVM standar dengan kernel tunggal.

4.1 *Dataset*

Dataset yang digunakan pada penelitian ini merupakan data citra berwarna yang berupa objek tunggal, yaitu citra dengan satu gambar objek yang dominan. Jumlah dari *dataset* citra tersebut adalah 2500 citra, yang terdiri dari 5 kategori objek yaitu citra pesawat terbang, motor, mobil, kucing dan citra wajah dimana masing-masing kategori terdiri dari 500 citra dengan pembagian 90% data untuk proses *training* dan 10% data untuk proses *testing*. Selain itu uji coba juga dilakukan untuk mengenali wajah seseorang, *dataset* yang digunakan untuk hal tersebut adalah *dataset* citra berupa wajah 10 orang dengan masing-masing berjumlah 20 citra dengan berbagai ekspresi dan latar belakang yang berbeda dengan pembagian 80% data untuk proses *training* dan 20% data untuk proses *testing*.

Untuk proses penambahan kelas, uji coba dilakukan dengan melakukan 3 kali penambahan, dimana penambahan kelas pertama ditambahkan kelas baru berupa citra bunga sebanyak 250 citra, kemudian penambahan kelas kedua ditambahkan kelas baru berupa citra burung dengan jumlah 150 citra, dan penambahan kelas ketiga berupa citra kupu-kupu dengan jumlah 100 citra dengan ketentuan pembagian yang sama yaitu 90% data untuk proses *training* dan 10% data untuk proses *testing*. Sedangkan untuk pengenalan wajah ditambahkan citra wajah baru sebanyak 15 citra untuk penambahan kelas pertama, 10 citra untuk penambahan kelas kedua dan 5 citra untuk penambahan kelas ketiga secara berturut-turut dengan ketentuan 80% data untuk proses *training* dan 20% data untuk proses *testing*. Gambar 4.1 dan 4.2 berikut ini adalah contoh dari citra yang digunakan pada penelitian ini.

Gambar 4.1. Contoh citra uji coba



Gambar 4.2. Contoh citra pengenalan wajah

Untuk validasi dari metode yang telah dibangun, digunakan metode *k-fold cross-validation* seperti yang telah dijelaskan sebelumnya pada Bab 3, dimana data dibagi menjadi sepuluh kelompok ($k = 10$) untuk klasifikasi objek umum dan 5 kelompok ($k = 5$) untuk pengenalan wajah, dengan pembagian data x berjumlah 9 kelompok untuk proses *training* dan y berjumlah 1 kelompok untuk *testing* serta 5 kelompok untuk *training* dan 1 kelompok untuk *testing* pada percobaan pengenalan

wajah. Kemudian percobaan dilakukan sebanyak 10 kali untuk klasifikasi objek umum dan 5 kali untuk pengenalan wajah, dengan data *training* dan *testing* yang selalu berbeda sesuai dengan skema yang ditunjukkan pada Gambar 3.4.

4.2 Ekstraksi Fitur

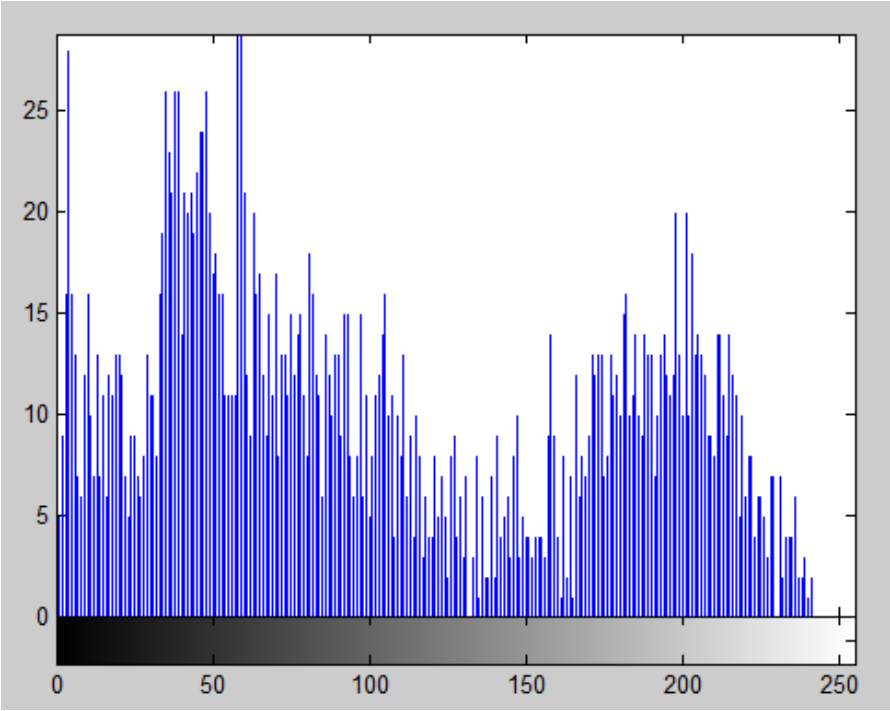
Langkah pertama yang dilakukan dalam proses implementasi ini adalah memproses data citra untuk mendapatkan nilai histogram dari citra tersebut. Histogram didapat dengan menghitung frekuensi kemunculan nisbi (*relative*) dari intensitas pada citra tersebut. Secara matematis histogram citra dapat dihitung sesuai dengan Persamaan (3.1). Jadi, misalkan terdapat sebuah citra *input* seperti Gambar 4.3 dibawah ini:

Gambar 4.3. Contoh citra *input*

Kemudian setelah di ubah menjadi *greyscale* dan dilakukan *resize* sehingga menjadi citra dengan ukuran 50 x 50 maka didapatkan matriks sebagai berikut

6	5	4	2	4	2	2	2	…	162
7	7	4	4	3	3	4	3	…	173
7	5	3	2	1	1	2	1	…	174
21	21	19	19	19	17	17	15	…	177
29	30	31	31	33	34	34	35	…	177
8	8	8	6	6	6	7	8	…	181
5	4	4	4	4	3	4	14	…	183
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
34	33	33	36	37	34	35	51	…	36

Dari matriks tersebut didapatkan histogram sebagai berikut:



Gambar 4.4. Contoh histogram

Gambar 4.4 menunjukkan histogram dari Gambar 4.3 dimana sumbu horizontal menunjukkan nilai *grey level* sedangkan sumbu vertikal menunjukkan nilai jumlah dari *grey level* tersebut. Dari hasil ini maka didapat vektor *input* untuk setiap citra berupa $X = (x_1, x_2, x_3, \dots, x_{256})$ dengan x_i merupakan *feature* yang didapat dari jumlah pada setiap *grey* level.

4.3 Proses Klasifikasi

Pertama, proses *training* SVM dilakukan dengan data citra berjumlah 2250 dengan 5 kategori berbeda yaitu citra pesawat, motor, mobil, kucing dan citra wajah. Pada proses *training* ini fungsi kernel yang digunakan pada pembelajaran SVM berbasis multi kernel merupakan fungsi kernel hasil kombinasi dari fungsi-fungsi kernel linier pada Persamaan (2.8), kernel Polynomial pada Persamaan (2.9) dengan order 3 dan kernel RBF pada Persamaan (2.10). Kombinasi dari fungsi-fungsi kernel tersebut dilakukan sesuai dengan rumusan pada Persamaan (2.12). Berikut adalah langkah untuk melakukan proses *training* dengan algoritma SVM.

Algoritma *training* SVM

1. Tentukan *input X* , target *Y* dan nilai penalti C
2. Tentukan fungsi Multi Kernel dari Persamaan (2.8), (2.9) dan (2.10) yang dikombinasikan sesuai dengan Persamaan (2.12)
3. Hitung matriks Kernel *K* yaitu perkalian *dot product* dari vektor *input* sesuai dengan fungsi Multi Kernel yang telah ditentukan.
4. Temukan solusi optimal untuk nilai α pada Persamaan (2.21).
5. Dapatkan data *support vector* yang memenuhi Persamaan (2.25)
6. Dapatkan bias yang diperoleh dari $b = y_i - wx_i$.

Karena SVM merupakan pengklasifikasi biner maka digunakan metode *one-against-all* dengan skema yang telah ditunjukkan pada Gambar 3.1 dimana proses *training* pertama dilakukan dengan memberikan target pada kelompok citra pesawat berupa $y = 1$ dan citra sisanya dengan $y = -1$ kemudian proses *training* selanjutnya melibatkan citra-citra yang sebelumnya dikategorikan pada kelas negatif dimana keempat citra tersebut juga dilakukan *training* dengan cara yang sama dengan proses *training* yang pertama, untuk lebih jelasnya bisa dilihat pada Tabel 4.1 di bawah ini:

Tabel 4.1 *Input* dan Target SVM1

Kelompok citra	***Input***	**Target**
Pesawat	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Motor	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Kucing	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Mobil	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.1 menunjukkan *input* dan target dari proses *training* SVM yang pertama, *Xi* menunjukkan vektor *input* $ke - i$ sedangkan Y menunjukkan nilai dari target kelas yang dipelajari, pada pembelajaran pertama ini, kelompok citra pesawat dianggap sebagai kelompok dari kelas positif atau target $Y = 1$, sedangkan sisanya dianggap sebagai kelas negatif atau target $Y = -1$.

Tabel 4.2 *Input* dan Target SVM2

Kelompok citra	***Input***	**Target**
Motor	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Kucing	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Mobil	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.2 menunjukkan *input* dan target pada proses *training* dari SVM kedua yang melibatkan kelompok-kelompok citra yang pada proses sebelumnya dianggap sebagai kelas negatif, disini kelompok motor dianggap sebagai kelas positif sedangkan kelompok citra lainnya dianggap sebagai kelompok dari kelas negatif.

Tabel 4.3 *Input* dan Target SVM3

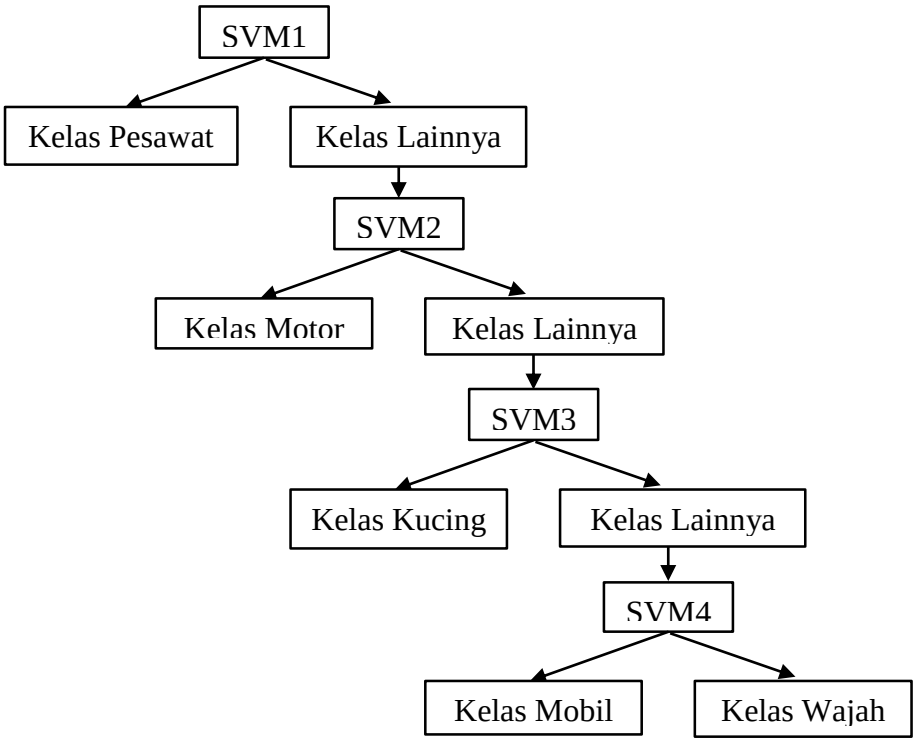
Kelompok citra	***Input***	**Target**
Kucing	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Mobil	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Pada *training* SVM yang ketiga atau model SVM3 seperti yang ditunjukkan pada Tabel 4.3, kelompok dari citra kucing dianggap sebagai kelas positif sedangkan kelompok citra mobil dan wajah merupakan kelompok dari kelas negatif.

Tabel 4.4 *Input* dan Target SVM4

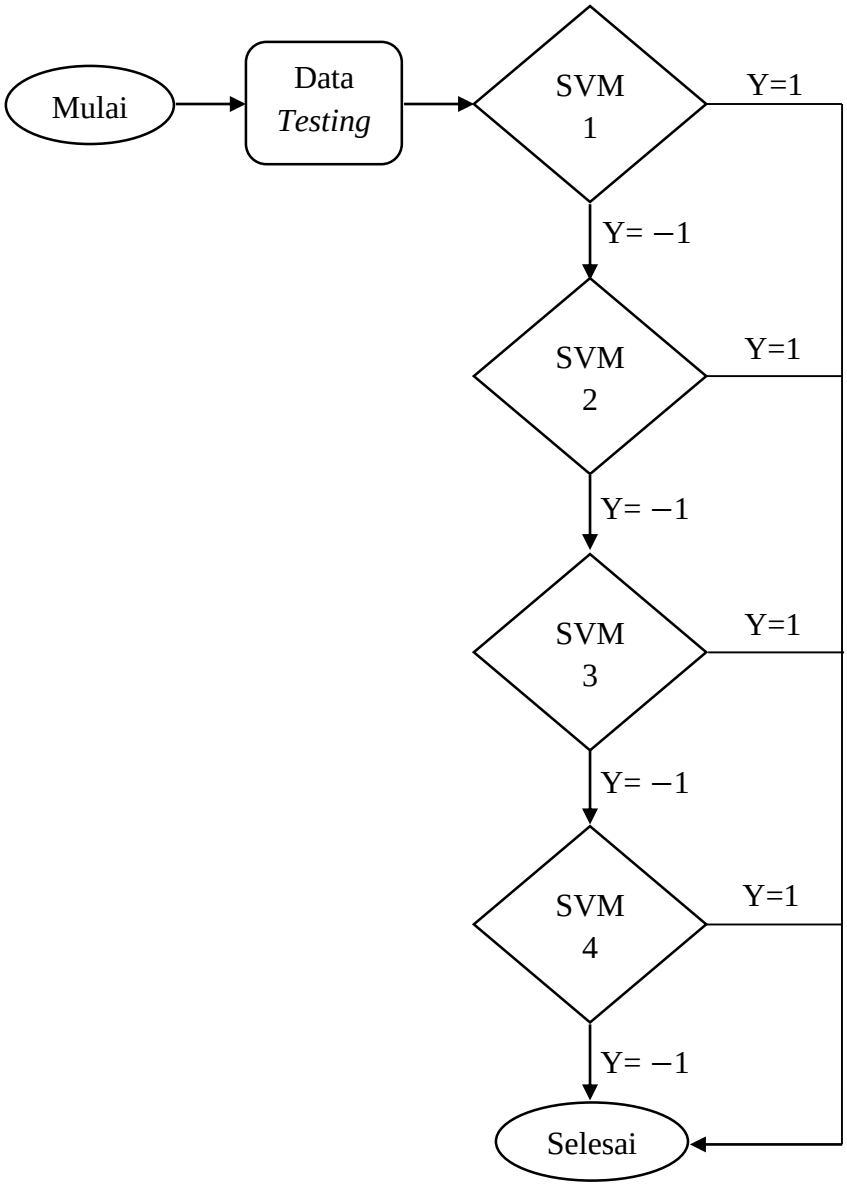
Kelompok citra	***Input***	**Target**
Mobil	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Wajah	$Xi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.4 menunjukkan bahwa pada *training* yang terakhir atau SVM4 tersisa dua kelompok citra yang pada proses-proses sebelumnya selalu dianggap sebagai kelas negatif. Pada proses *training* ini, kedua kelompok citra mobil diberikan target $Y = 1$ dan kelompok citra wajah diberikan target $Y = -1$. Jika digambarkan sesuai dengan skema pada Gambar 3.1 maka proses *training* pada 5 kelas awal ini dapat digambarkan sebagaimana Gambar 4.5 berikut ini:



Gambar 4.5. Skema *training* SVM pada 5 kelas awal

Selanjutnya, untuk proses *testing* dilakukan dengan skema dan urutan yang sama dengan proses *training* yang telah dilakukan yaitu dengan mencari hasil klasifikasi dari model yang terbentuk secara berurutan dari model SVM1 sampai model terakhir atau SVM4, hingga didapatkan kelas yang tepat, untuk lebih jelasnya dapat dilihat pada diagram alir pada Gambar 4.6 dibawah ini:



Gambar 4.6. Diagram alir *testing* SVM *One-Againts-All*

Pada diagram alir proses *testing* yang ditunjukkan oleh Gambar 4.6, data *testing* pertama kali diuji pada model SVM1 jika hasilnya menunjukkan $Y = 1$ maka proses *testing* berhenti dan disimpulkan bahwa hasil klasifikasi citra dari data *testing* tersebut adalah kelas positif (kelompok citra pesawat) namun jika hasilnya menunjukkan $Y = -1$ maka data tersebut masih memiliki kemungkinan berada pada keempat kelas yang tersisa, sehingga proses *testing* dilanjutkan pada model kedua, ketiga, atau keempat sampai ditemukan hasil dari kelas yang sesuai.

Hal serupa juga dilakukan pada proses *training* untuk pengenalan wajah, dimana pada proses awal dilakukan *training* sebanyak 10 kelas dengan total 200 data citra. Sedangkan proses *testing* juga dilakukan dengan langkah yang sama seperti yang ditunjukkan pada Gambar 4.6, sebanyak dengan model SVM yang terbentuk yaitu 9 model SVM. Kemudian, untuk proses penambahan kelas dilakukan dengan tiga tahap dimana masing-masing tahap ditambahkan satu kelas baru dengan ketentuan data tambah seperti yang telah dijelaskan pada Subbab 4.1.

4.4 Proses Klasifikasi dengan Pembelajaran yang Bertambah

Menurut Zhang dkk (2009), setelah *training* SVM dijalankan maka didapatkan beberapa data x_i yang memenuhi $\alpha_i = 0$ dimana data tersebut merupakan data yang berada diluar dari *margin*, serta beberapa data x_i yang memenuhi $0 \leq \alpha_i \leq C$ merupakan data yang tepat berada pada bidang pembatas $f(x) = \pm 1$ (lihat Gambar 2.1), sedangkan data dengan $C = \alpha$ adalah data yang berada didalam *margin*, atau dapat dituliskan sebagai berikut:

$$\alpha_i = 0 \rightarrow |f(x_i)| \geq 1$$
$$0 \leq \alpha_i \leq C \rightarrow |f(x_i)| = 1$$
$$\alpha = C \rightarrow |f(x_i)| \leq 1$$

Dimana setiap data yang berada pada bidang pembatas $f(x) = \pm 1$ adalah data yang disebut dengan *support vector*.

Hal pertama yang perlu diperhatikan dalam pembelajaran yang bertambah adalah menentukan data yang digunakan dalam *training* SVM berikutnya yang mana data tersebut merupakan data *support vector* dari hasil *training* SVM sebelumnya. Data yang demikianlah yang nantinya digabungkan dengan data baru

dan digunakan untuk proses *training* untuk mendapatkan kelas baru. Pembelajaran yang bertambah atau *incremental learning* merupakan proses pembelajaran yang iteratif. Data baru yang terbentuk diproses dengan algoritma SVM sehingga didapatkan kembali *support vector* dan *hyperplane* baru dimana kemudian hasil dari proses *training* tersebut juga digunakan untuk proses *training* berikutnya dan begitu seterusnya.

Jadi, misalkan *dataset* lama adalah x_a dan *dataset* baru adalah x_b, langkah dari proses pembelajaran yang bertambah ini adalah sebagai berikut:

1. Lakukan *training* SVM dengan data x_a
2. Dapatkan *support vector* x_a yang memenuhi $f(x) = \pm 1$
3. Tambahkan data x_b sehingga data $x_a = x_a \cup x_b$
4. Ulangi langkah 1.

Untuk tahap penambahan kelas yang pertama, ditambah satu kelas baru pada model SVM 5 kelas yang telah dibentuk sebelumnya, data baru yang ditambahkan adalah data kelompok citra bunga, dengan penambahan satu kelas ini maka model SVM baru yang terbentuk bertambah pula, berikut adalah pembagian *input* dan target pada model SVM yang dibentuk pada proses *training* yang baru:

Tabel 4.5 *Input* dan Target SVM1[1]

Kelompok citra	***Input***	**Target**
Pesawat	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Motor	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Kucing	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Mobil	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Bunga	$X = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Pada fase pembelajaran yang bertambah ini, model SVM pertama disimbolkan dengan SVM1[1] dengan pembagian *input* dan target yang ditunjukkan oleh Tabel 4.5. Pada Tabel 4.5, $XSVi$ merupakan vektor *input* dari data ke-i yang merupakan

support vector dari pembelajaran SVM1 sedangkan vektor *input* dari kelompok citra bunga (kelas baru yang ditambahkan) tetap disimbolkan dengan X.

Tabel 4.6 *Input* dan Target SVM2[1]

Kelompok citra	***Input***	**Target**
Motor	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Kucing	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Mobil	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Bunga	$X = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.6 menunjukkan *input* dan target dari model SVM kedua dari proses *training* SVM dengan pembelajaran yang bertambah, $XSVi$ pada kelompok citra motor sampai dengan wajah merupakan *support vector* dari pembelajaran SVM2

Tabel 4.7 *Input* dan Target SVM3[1]

Kelompok citra	***Input***	**Target**
Kucing	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Mobil	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Wajah	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Bunga	$X = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.7 menunjukkan pembagian *input* dan target dari model SVM yang ketiga dengan $XSVi$ merupakan *support vector* dari pembelajaran SVM3

Tabel 4.8 *Input* dan Target SVM4[1]

Kelompok citra	***Input***	**Target**
Mobil	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Wajah	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$
Bunga	$X = (x_1, x_2, x_3, \dots, x_{256})$	$Y = -1$

Tabel 4.8 menunjukkan pembagian *input* dan target dari model SVM yang keempat dengan $XSVi$ merupakan *support vector* dari pembelajaran SVM4

Tabel 4.9 *Input* dan Target SVM5[1]

Kelompok citra	***Input***	**Target**
Wajah	$XSVi = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 1$
Bunga	$X = (x_1, x_2, x_3, \dots, x_{256})$	$Y = 0$

Pada proses *training* model SVM yang terakhir, data *input* $XSVi$ yang ditunjukkan pada Tabel 4.9 merupakan data dari kelompok citra wajah yang merupakan data *support vector* yang didapat dari pembelajaran SVM4. Setelah semua *training* dilakukan maka didapat 5 model SVM baru hasil klasifikasi 6 kelas. Untuk proses *testing* dapat dilakukan dengan cara yang sama seperti yang dilakukan pada proses *testing* terhadap model sebelumnya.

Proses yang sama juga berlaku untuk setiap penambahan kelas berikutnya, sehingga pada akhirnya terbentuk 8 kelas hasil dari tiga kali penambahan kelas secara berturut-turut. Hal yang sama juga dilakukan dalam mengklasifikasikan *dataset* wajah untuk uji coba pengenalan wajah. Dengan jumlah kelas yang pada awalnya 10 maka pada akhirnya terbentuk 13 kelas setelah mengalami tiga kali penambahan kelas.

4.5 Hasil dan Pembahasan

Uji coba pada penelitian ini dilakukan dengan menggunakan beberapa perangkat keras dan perangkat lunak komputer. Dengan spesifikasi lingkungan perancangan sistem dapat dilihat pada Tabel 4.10.

Tabel 4.10 Spesifikasi Lingkungan Perancangan Sistem

Perangkat Keras	Prosesor	:	Intel® Core™ i5-5200U CPU @ 2,20 GHz
	Memory	:	4 GB DDR4
Perangkat Lunak	Sistem Operasi	:	Windows 8.1 Pro
	Tools	:	MatlabR2012b x64

Seperti yang dijelaskan sebelumnya bahwa tujuan utama dari *image retrieval* atau temu kembali citra adalah mencari citra-citra yang memiliki kemiripan karakteristik dari citra acuan (*input*). Sesuai dengan diagram alir pada Gambar 3.3, proses pencarian ini dilakukan dengan membandingkan hasil dari *testing* citra *input* maupun citra dari *database testing* terhadap model SVM hasil dari proses *training* yang telah dilakukan sebelumnya. Jika hasil *testing* menunjukkan bahwa citra dari *database testing* yang sedang diproses berada pada kelas yang sama dengan hasil dari *testing* citra *input*, maka citra tersebut ditampilkan sebagai hasil dari pencarian yang berarti citra tersebut memiliki kemiripan karakteristik dengan citra *input*. Sedangkan jika hasil perbandingan menunjukkan kelas yang berbeda maka citra tersebut tidak ditampilkan sebagai hasil pencarian atau dengan kata lain citra tersebut tidak sesuai dengan citra yang dicari oleh *user*. Gambar 4.7 berikut adalah *user interface* dari aplikasi temu kembali citra yang telah dibangun:

Gambar 4.7. *User interface* dari temu kembali citra

Selanjutnya, untuk mengukur performa dari temu kembali citra yang telah dibangun, maka dilakukan perhitungan *precision*, *recall* dan akurasi sesuai dengan rumusan pada Persamaan (3.2), (3.3), dan (3.4). Selain itu, juga dilakukan perhitungan waktu komputasi selama proses *training* maupun proses *testing* yang dilakukan oleh sistem dengan satuan detik.

4.5.1 Klasifikasi Objek Umum

Pada subbab ini dibahas hasil dari uji coba klasifikasi dengan 5 objek umum. Tabel 4.11 berikut merupakan hasil dari pengujian sistem hasil klasifikasi SVM dengan multi kernel 5 kelas yang masing-masing kelas terdiri dari 450 citra untuk data *training* dan 50 untuk *testing* (Hasil rata-rata dari seluruh percobaan):

Tabel 4.11 Tabel hasil klasifikasi dengan 5 kelas

SVM		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Pesawat	50	19	10	29	9.414	66%	38%	84%
Motor	50	37	17	54	10.718	**69%**	**74%**	**88%**
Kucing	50	16	30	46	10.499	35%	32%	74%
Mobil	50	25	30	55	11.411	45%	50%	78%
Wajah	50	34	32	66	11.080	52%	68%	81%
Rata-Rata					10.624	53%	52%	81%
Waktu *Training*	10.148							

Hasil percobaan ini menunjukkan bahwa klasifikasi dengan SVM multi kernel memiliki nilai akurasi yang baik dengan nilai 81%, sedangkan nilai *recall* dan *precision* masing-masing sebesar 52% dan 53%. Pada percobaan ini kelas motor memiliki nilai *precision, recall* dan akurasi yang paling tinggi diantara kelas lainnya, hal ini dikarenakan sebagian besar citra dari kelas ini memiliki *background* yang sama dan satu warna yaitu putih, daripada citra lainnya yang cenderung memiliki *background* yang lebih bervariasi, sehingga kelompok citra motor ini lebih mudah dikenali sebagai satu kelas daripada yang lainnya. Sebaliknya kelompok citra kucing merupakan kelompok yang memiliki nilai rata-rata *precision, recall* maupun akurasi yang paling buruk daripada kelompok lainnya, hal ini dikarenakan banyaknya *background* dari kelompok citra ini yang memiliki kemiripan dengan *background* dari citra lainnya, sehingga citra ini lebih sulit dibedakan dengan kelompok-kelompok lainnya. Jika dilihat dari sisi waktu komputasinya, pembelajaran SVM multi kernel ini memiliki waktu yang cepat

dengan 10.148 detik dimana sistem ini melakukan 4 kali proses *training* untuk sekali proses klasifikasi, artinya dibutuhkan waktu sekitar 2.5 detik untuk setiap *training*, sedangkan untuk waktu *testing* sistem ini membutuhkan waktu rata-rata 10.624 detik.

Selanjutnya, Tabel 4.12 menunjukkan hasil dari percobaan sistem dengan 6 kelas yang merupakan hasil pembelajaran bertambah dari 5 kelas sebelumnya. Sistem yang sebelumnya terdiri dari 5 kelas kemudian ditambahkan satu kelas baru berupa kelompok citra bunga yang terdiri dari 225 data *training* dan 25 data *testing*. Berikut adalah hasil dari percobaan tersebut:

Tabel 4.12 Tabel hasil klasifikasi penambahan kelas pertama

SVM Update 1		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Pesawat	50	22	16	38	10.622	58%	44%	84%
Motor	50	33	18	51	11.128	65%	66%	87%
Kucing	50	9	26	35	10.778	26%	18%	76%
Mobil	50	17	24	41	11.061	41%	34%	79%
Wajah	50	27	22	49	11.082	55%	54%	84%
Bunga	25	17	44	61	11.691	28%	68%	81%
Rata-rata					11.061	45%	47%	82%
Waktu *Training*	8.126							

Dilihat dari hasil yang ditunjukkan pada Tabel 4.12, terlihat bahwa meskipun kelas pada sistem bertambah dari 5 kelas menjadi 6 kelas yang artinya pada bagian ini sistem melakukan proses *training* satu kali lebih banyak daripada sebelumnya, waktu yang dibutuhkan selama proses klasifikasi ternyata lebih cepat 2.002 detik daripada proses sebelumnya. Hal ini dikarenakan, data yang dipakai selama proses *training* kali ini merupakan data *support vector* yang memiliki jumlah lebih sedikit daripada jumlah data awal yang dipakai sistem saat mengklasifikasikan 5 kelas.

Jika dilihat dari performanya, baik nilai *precision*, maupun *recall* serta akurasi dari percobaan kali ini juga tidak banyak berubah, bahkan memiliki nilai akurasi yang lebih baik 1% meskipun nilai *precision* dan *recall* berkurang masing-masing

8% dan 5%, hal ini menunjukkan bahwa pembelajaran secara bertambah ini masih memiliki performa yang baik meskipun proses *training* tidak melibatkan semua data awal seperti pada proses sebelumnya.

Percobaan selanjutnya dilakukan dengan menambahkan lagi satu kelas baru pada sistem yang sebelumnya, sehingga jumlah kelas sekarang adalah 7 kelas. Penambahan kali ini dilakukan dengan menambahkan citra burung sebanyak 135 untuk *training* dan 15 data untuk proses *testing*. Hasil dari percobaan ditunjukkan pada Tabel 4.13 berikut ini:

Tabel 4.13 Tabel hasil klasifikasi penambahan kelas kedua

SVM Update 2		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Pesawat	50	22	21	43	11.489	51%	44%	83%
Motor	50	34	19	53	13.473	64%	68%	88%
Kucing	50	10	22	32	10.480	31%	20%	79%
Mobil	50	17	27	44	12.643	39%	34%	79%
Wajah	50	26	22	48	14.350	54%	52%	84%
Bunga	25	16	43	59	12.584	27%	64%	82%
Burung	15	4	7	11	10.836	36%	27%	94%
Rata-rata					12.265	43%	44%	84%
Waktu *Training*	8.912							

Setelah penambahan satu kelas lagi, sistem masih menunjukkan performa yang baik walaupun nilai *recall* dan *precision* turun sebesar 3% dan 2% dari sebelumnya, namun nilai akurasi dari sistem mengalami kenaikan sebesar 2%. Hal yang perlu diperhatikan adalah waktu komputasinya, meskipun jumlah kelas bertambah dari 6 kelas menjadi 7 kelas yang berarti sistem melakukan *training* satu kali lebih banyak daripada sebelumnya, namun waktu yang dibutuhkan selama proses klasifikasi berlangsung hanya 8.912 detik, lebih cepat 1.236 detik bila dibandingkan proses pertama, yang mana ini berarti sistem kali ini melakukan *training* dua kali lebih banyak dari sistem saat mengklasifikasikan 5 kelas.

Selanjutnya, dilakukan penambahan lagi satu kelas pada sistem sebelumnya, berupa citra kupu-kupu berjumlah 90 untuk proses *training* dan 10 untuk *testing*, sehingga nantinya didapatkan sistem dengan total 8 kelas dengan jumlah data secara keseluruhan dari awal yang terlibat adalah 3000 data citra. Hasil dari percobaan ini disajikan pada Tabel 4.14 berikut ini:

Tabel 4.14 Tabel hasil klasifikasi penambahan kelas ketiga

SVM Update 3		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Pesawat	50	20	21	41	12.223	49%	40%	83%
Motor	50	27	22	49	12.434	55%	54%	85%
Kucing	50	7	23	30	11.730	23%	14%	78%
Mobil	50	13	23	36	12.134	36%	26%	80%
Wajah	50	27	29	56	13.089	48%	54%	83%
Bunga	25	16	43	59	13.341	27%	64%	83%
Burung	15	3	6	9	10.609	33%	20%	94%
Kupu-kupu	10	1	19	20	11.189	5%	10%	91%
Rata-rata					12.094	35%	35%	85%
Waktu *Training*	9.161							

Pada percobaan kali ini, performa sistem mengalami penurunan yang cukup besar dari sebelumnya, dimana nilai *precision* dan *recall* menurun sampai dengan 8% dan 9%. Penurunan nilai rata-rata ini selain karena memang performa yang sedikit menurun seperti pada proses klasifikasi sebelumnya, juga dikarenakan nilai dari *precision* dan *recall* dari kelas baru yaitu kupu-kupu yang buruk yaitu hanya bernilai 5% dan 10%. Artinya, untuk kali ini sistem hanya mampu mengklasifikasikan kelas baru tersebut dengan baik dengan hanya mengenali satu data positif dengan benar dari 10 data positif yang di*training*kan dan mengenali 19 dari 281 data yang sebenarnya adalah data negatif sebagai data positif (untuk lebih jelasnya dapat dilihat pada *confusion matriks* dari kelompok kupu-kupu yang ditunjukkan oleh Tabel 4.15). Jika dilihat dari waktu komputasinya, waktu *training* sistem kali ini meningkat 0.249 detik dari sistem sebelumnya, namun waktu ini

masih lebih cepat 0.987 detik dari yang dilakukan sistem pada saat mengklasifikasikan 5 kelas.

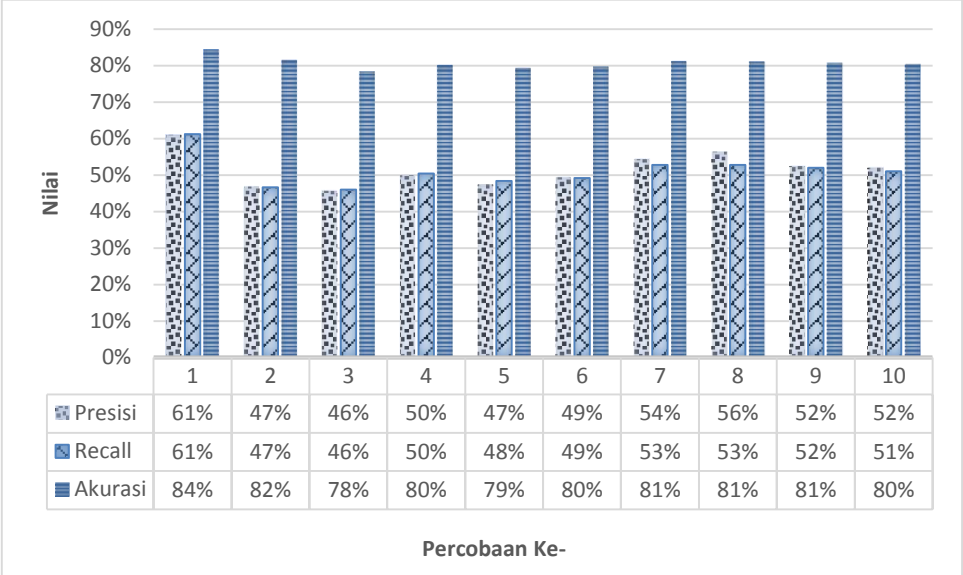
Tabel 4.15 *Confusion Matriks* hasil klasifikasi kelompok citra kupu-kupu

		Nilai Prediksi	
		True	***False***
Nilai Sebenarnya	***True***	1	9
	False	19	281

Dari hasil yang telah dibahas terlihat bahwa metode SVM multi kernel memiliki nilai akurasi yang baik yaitu 81% dan kemudian meningkat berturut-turut dari penambahan pertama sampai ketiga dengan nilai 82%, 84% dan 85%, sedangkan nilai *precision* untuk pembelajaran multi kernel mencapai 53% dan kemudian menurun secara teratur menjadi 45%, 43% dan 35% selama tiga kali penambahan kelas. Hal ini juga terjadi pada nilai *recall* dimana yang semula nilainya 52% mengalami penurunan menjadi 47%, 44% dan 35% untuk tiga kali penambahan kelas secara berturu-turut.

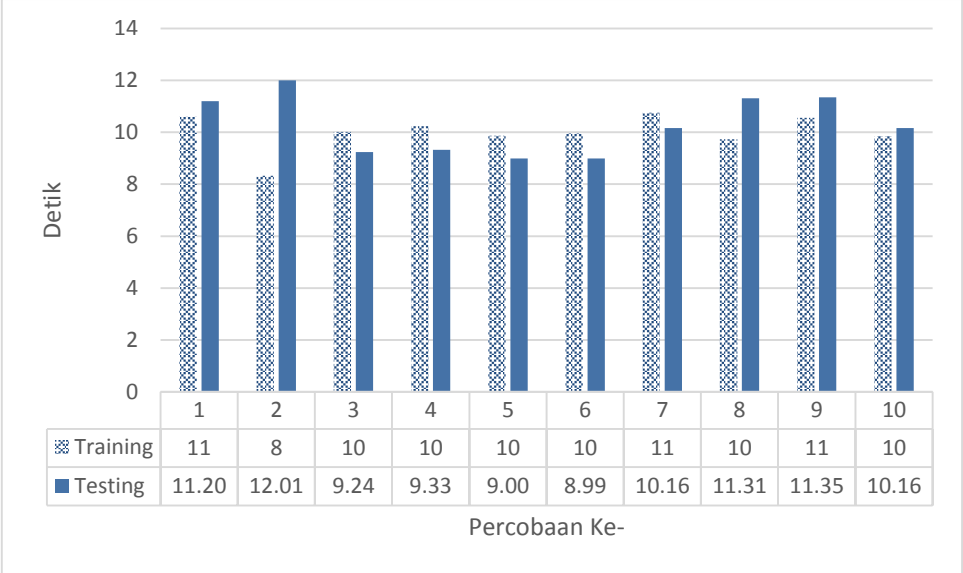
Jika dilihat dari segi waktu komputasi sistem pada pengklasifikasian objek umum, walaupun jumlah kelas yang diklasifikasikan lebih banyak dimana artinya dengan metode *one-against-all* proses *training* SVM yang dijalankan pun semakin banyak, namun sistem menunjukkan bahwa waktu komputasi yang diperlukan lebih cepat, dimana pada awalnya sistem melakukan *training* untuk 5 kelas membutuhkan waktu 10.148 sedangkan selama penambahan kelas sistem hanya membutuhkan waktu selama 8.126 untuk *training* 6 kelas, 8.912 untuk 7 kelas dan 9.161 untuk 8 kelas.

Untuk lebih jelas mengetahui performa dari sistem ini, grafik dari rata-rata hasil *precision*, *recall*, akurasi dan waktu komputasi secara menyeluruh untuk semua kelas pada setiap percobaan yang dilakukan secara *cross validation* disajikan pada Gambar 4.8 sampai dengan Gambar 4.15:



	1	2	3	4	5	6	7	8	9	10
Presisi	61%	47%	46%	50%	47%	49%	54%	56%	52%	52%
Recall	61%	47%	46%	50%	48%	49%	53%	53%	52%	51%
Akurasi	84%	82%	78%	80%	79%	80%	81%	81%	81%	80%

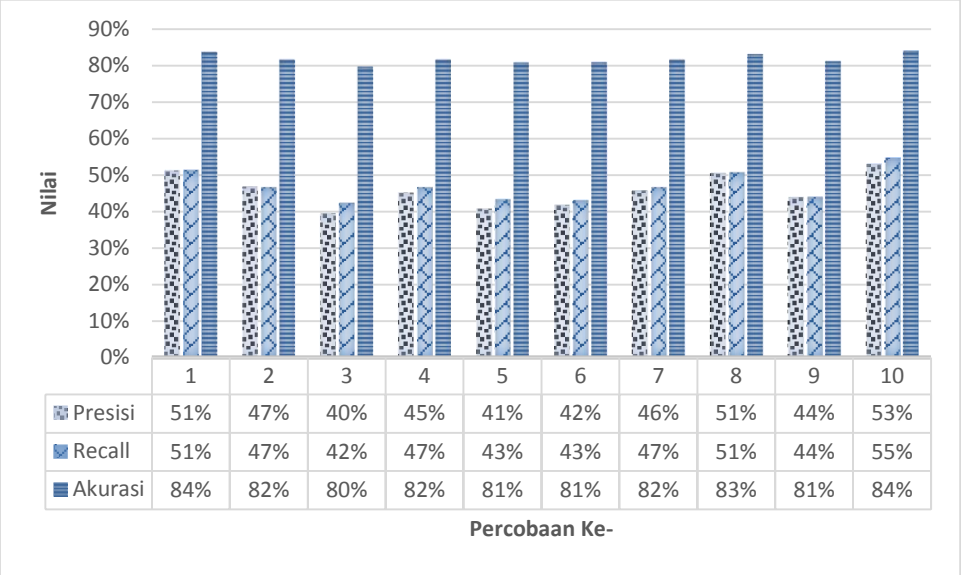
Gambar 4.8. Nilai *precision*, *recall*, akurasi 5 kelas



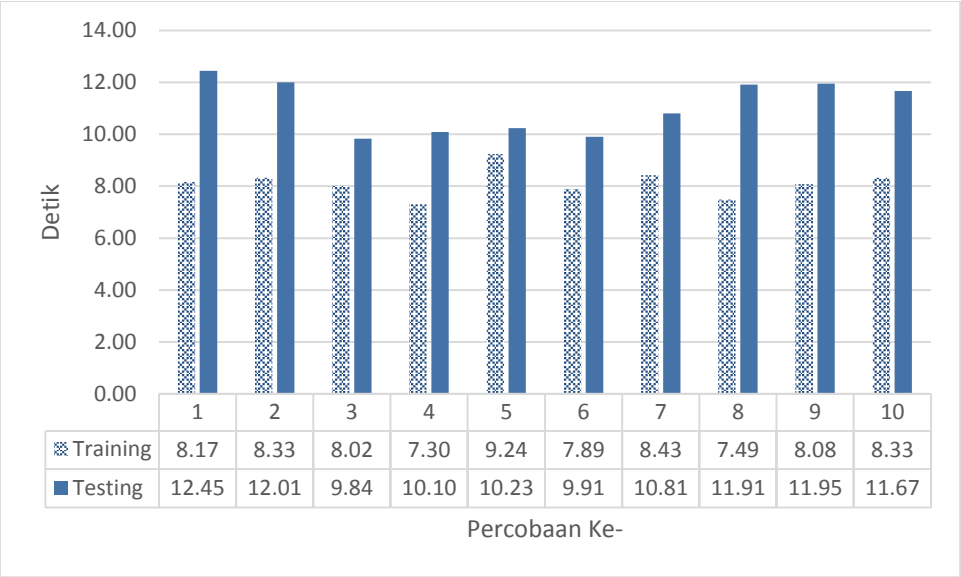
	1	2	3	4	5	6	7	8	9	10
Training	11	8	10	10	10	10	11	10	11	10
Testing	11.20	12.01	9.24	9.33	9.00	8.99	10.16	11.31	11.35	10.16

Gambar 4.9. Waktu komputasi 5 kelas

Pada Gambar 4.8 dan 4.9 terlihat bahwa pembelajaran SVM multi kernel dengan 5 kelas memiliki performa yang baik dalam nilai *precision*, *recall* maupun akurasi serta waktu komputasinya dan juga cenderung konstan pada tiap percobaan dimana tidak ada penurunan atau kenaikan nilai yang besar antara percobaan satu dengan yang lainnya.



Gambar 4.10. Nilai *precision*, *recall*, akurasi penambahan kelas pertama

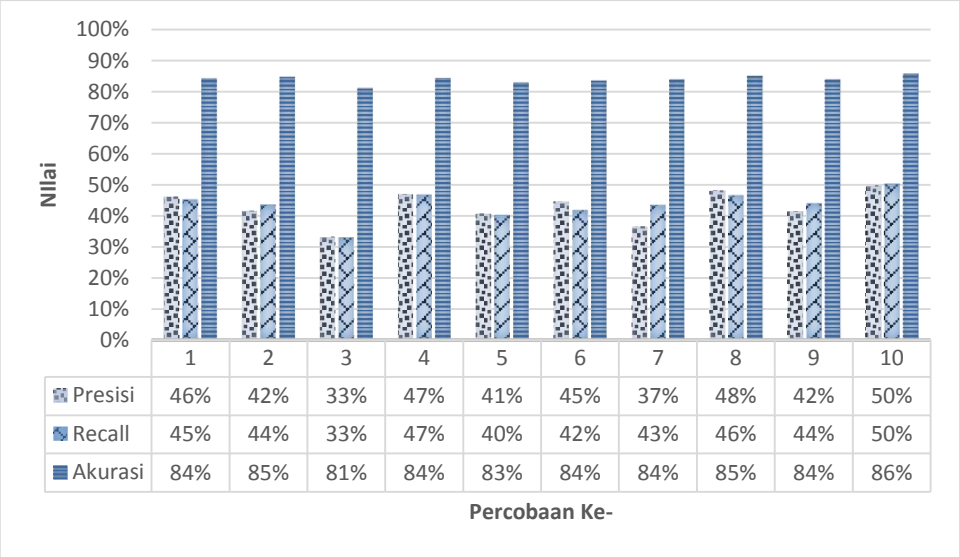


Gambar 4.11. Waktu komputasi penambahan kelas pertama

Gambar 4.10 dan Gambar 4.11 menunjukkan grafik performa dan waktu komputasi dari keseluruhan percobaan untuk sistem dengan pembelajaran yang bertambah, yaitu dengan penambahan satu kelas pada sistem sebelumnya yang terdiri dari 5 kelas. Pada grafik yang ditunjukkan oleh Gambar 4.10 dan Gambar 4.11 juga terlihat bahwa SVM multi kernel dengan pembelajaran bertambah memiliki performa yang cenderung konstan dari setiap percobaan yang dilakukan

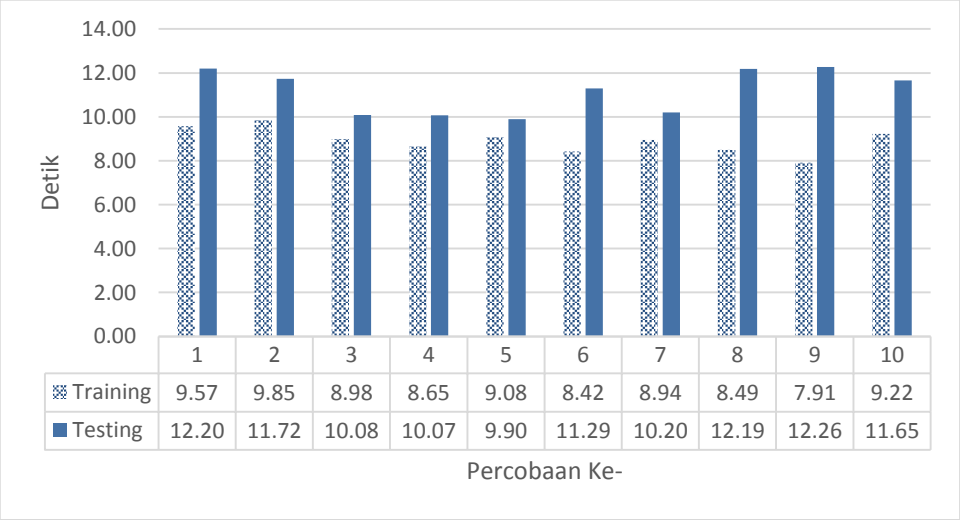
seperti pada sistem dengan pembelajaran multi kernel sebelumnya meskipun nilai *precision* dan *recall* dari sistem ini sedikit lebih rendah dari nilai pada sistem sebelumnya.

Kemudian, Gambar 4.12 dan Gambar 4.13 adalah grafik yang menunjukkan performa dan waktu komputasi hasil dari keseluruhan percobaan dari sistem pada saat penambahan kelas yang kedua.



	1	2	3	4	5	6	7	8	9	10
Presisi	46%	42%	33%	47%	41%	45%	37%	48%	42%	50%
Recall	45%	44%	33%	47%	40%	42%	43%	46%	44%	50%
Akurasi	84%	85%	81%	84%	83%	84%	84%	85%	84%	86%

Gambar 4.12. Nilai *precision*, *recall*, akurasi penambahan kelas kedua

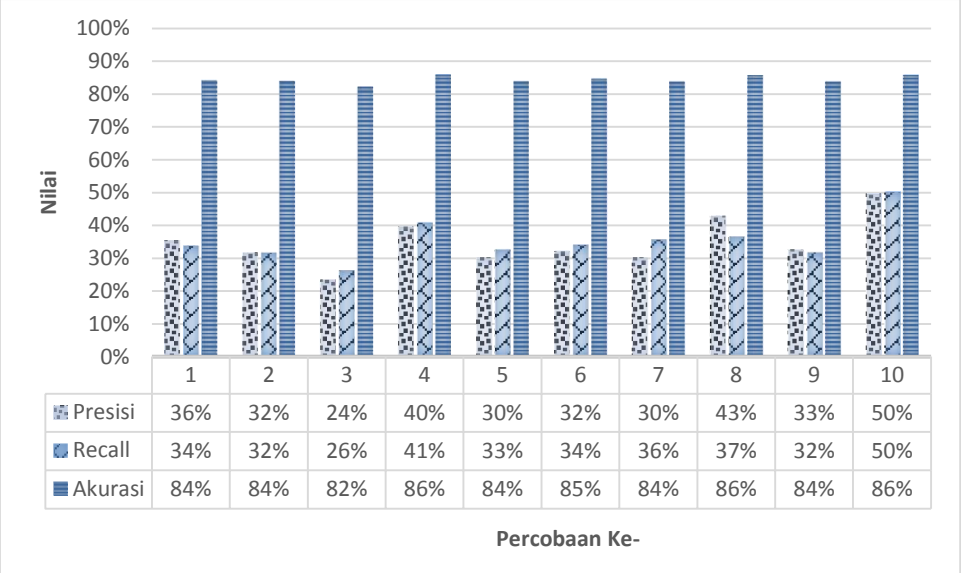


	1	2	3	4	5	6	7	8	9	10
Training	9.57	9.85	8.98	8.65	9.08	8.42	8.94	8.49	7.91	9.22
Testing	12.20	11.72	10.08	10.07	9.90	11.29	10.20	12.19	12.26	11.65

Gambar 4.13. Waktu komputasi akurasi penambahan kelas kedua

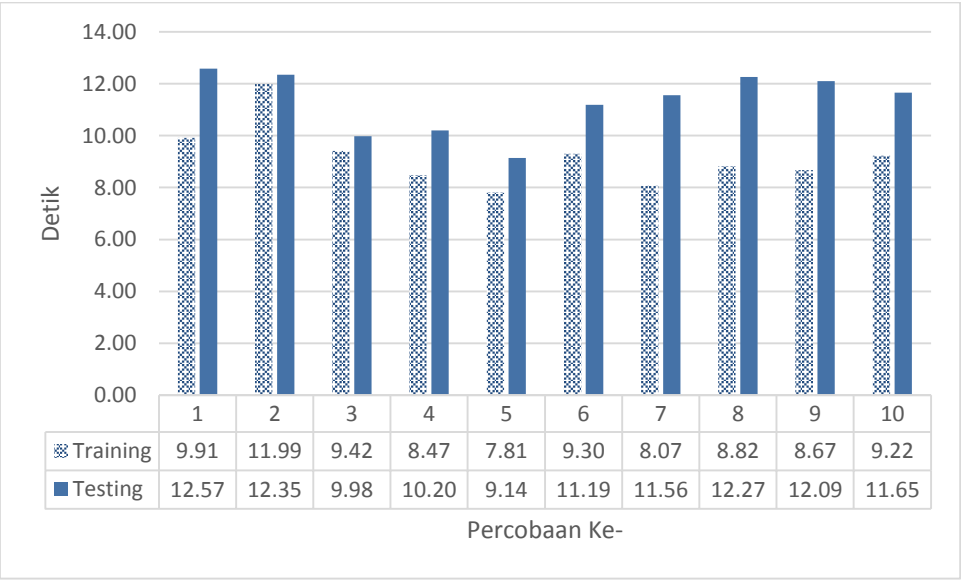
Seperti halnya penambahan kelas pertama, pada penambahan kedua kali ini sistem masih menunjukkan performa yang sama konstannya pada tiap percobaan dengan performa dan waktu komputasi pada sistem sebelumnya.

Gambar 4.14 dan 4.15 berikut ini adalah grafik yang menunjukkan performa dan waktu komputasi dari sistem pada penambahan kelas yang ketiga.



	1	2	3	4	5	6	7	8	9	10
Presisi	36%	32%	24%	40%	30%	32%	30%	43%	33%	50%
Recall	34%	32%	26%	41%	33%	34%	36%	37%	32%	50%
Akurasi	84%	84%	82%	86%	84%	85%	84%	86%	84%	86%

Gambar 4.14. Nilai *precision*, *recall*, akurasi penambahan kelas ketiga



	1	2	3	4	5	6	7	8	9	10
Training	9.91	11.99	9.42	8.47	7.81	9.30	8.07	8.82	8.67	9.22
Testing	12.57	12.35	9.98	10.20	9.14	11.19	11.56	12.27	12.09	11.65

Gambar 4.15. Waktu komputasi penambahan kelas ketiga

Pada Gambar 4.14 dan 4.15 terlihat bahwa pada penambahan satu kelas pada sistem untuk ketiga kalinya juga memperlihatkan performa yang konstan dari percobaan pertama hingga terakhir. Berdasarkan Gambar 4.8 sampai dengan Gambar 4.15 terlihat bahwa baik percobaan dengan 5 kelas awal maupun penambahan satu kelas dari yang pertama sampai yang ketiga memiliki pola kenaikan dan penurunan nilai yang sama dari percobaan satu ke percobaan lainnya. Hal ini menunjukkan bahwa pembelajaran yang bertambah tidak mengubah performa dari sistem secara keseluruhan, walaupun mungkin nilainya sedikit menurun. Namun secara umum, dari hasil percobaan cukup membuktikan bahwa secara empiris, jika dilihat dari rata-rata nilai *precision*, *recall* dan akurasinya metode klasifikasi dengan pembelajaran yang bertambah ini, memiliki performa yang baik selama proses penambahan kelas dengan waktu komputasi yang lebih cepat.

4.5.2 Klasifikasi Pengenalan Wajah

Untuk hasil dari pengenalan wajah pada metode SVM dengan multi kernel, Tabel 4.16 berikut menyajikan hasil rata-rata yang didapat dari keseluruhan percobaan dari klasifikasi 10 tipe wajah dengan jumlah data *training* masing-masing sebanyak 16 citra dan data *testing* sebanyak 4 citra:

Tabel 4.16 Tabel hasil pengenalan wajah dengan 10 kelas

SVM		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Wajah 1	4	2	1	3	1.481	67%	50%	93%
Wajah 2	4	2	2	4	1.377	28%	50%	90%
Wajah 3	4	1	1	2	1.276	50%	25%	90%
Wajah 4	4	1	1	2	1.246	50%	25%	90%
Wajah 5	4	1	1	2	0.746	36%	25%	90%
Wajah 6	4	1	2	3	1.438	33%	25%	88%
Wajah 7	4	2	1	3	1.371	67%	50%	93%
Wajah 8	4	1	1	2	1.224	50%	25%	90%
Wajah 9	4	3	7	9	1.874	33%	75%	83%
Wajah 10	4	3	10	10	1.651	30%	75%	80%
Rata-rata					1.368	38%	43%	89%
Waktu *Training*	5.993							

Pada percobaan pengenalan wajah ini, penamaan wajah 1 sampai wajah 10 mengacu pada tipe wajah pada Gambar 4.2 secara terurut dari kanan ke kiri. Jika dilihat dari tabel hasil tersebut, dapat disimpulkan bahwa sistem juga mampu bekerja dengan baik jika digunakan untuk pengenalan wajah dimana didapatkan hasil akurasi mencapai 89% dan *precision* serta *recall* sebesar 38% dan 43%. Jika dilihat dari waktu komputasi, metode SVM multi kernel ini membutuhkan waktu sekitar 5.993 detik untuk melakukan sembilan kali proses *training* secara berurutan dengan jumlah data *training* sebanyak 160 citra, artinya sistem hanya membutuhkan waktu sekitar 0.665 detik untuk setiap proses *training*.

Kemudian Tabel 4.17 berikut menunjukkan hasil rata-rata dari pembelajaran bertambah pada sistem sebelumnya, sehingga kali ini sistem mengklasifikasikan data wajah menjadi 11 kelas, dengan penambahan satu kelompok tipe wajah ke-11 sebanyak 15 data dengan 12 citra untuk proses pembelajaran dan 3 citra untuk proses *testing*.

Tabel 4.17 Tabel hasil pengenalan wajah penambahan kelas pertama

SVM Update 1		Hasil Pencarian		Jml	Hasil			
Jml Sebenanya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Wajah 1	4	2	1	3	0.759	67%	50%	93%
Wajah 2	4	1	2	3	1.245	33%	25%	88%
Wajah 3	4	1	1	2	1.359	50%	25%	91%
Wajah 4	4	1	1	2	1.471	50%	25%	91%
Wajah 5	4	1	1	2	1.254	50%	25%	91%
Wajah 6	4	2	1	3	1.367	67%	50%	93%
Wajah 7	4	1	2	3	1.370	33%	25%	88%
Wajah 8	4	1	1	2	1.241	50%	25%	91%
Wajah 9	4	3	7	10	1.739	30%	75%	81%
Wajah 10	4	2	7	9	1.305	22%	50%	79%
Wajah 11	3	1	3	4	1.345	25%	33%	88%
Rata-rata					1.314	43%	37%	89%
Waktu *Training*	6.494							

Seperti halnya percobaan pada objek umum, setelah penambahan satu kelas ternyata sistem tetap memiliki performa yang baik, meskipun nilai *recall* berkurang 6% namun nilai akurasi masih tetap dengan 89% dan bahkan nilai *precision* naik sebesar 5%, dilihat dari waktu selama proses *training*, untuk melakukan *training* satu kali lebih banyak dibutuhkan waktu 0.501 detik lebih lama dari yang dilakukan sistem sebelumnya.

Pada percobaan berikutnya, ditambahkan satu kelas baru sebanyak 10 citra, dengan 8 citra untuk proses *training* dan 2 citra untuk proses *testing*. Hasilnya disajikan pada Tabel 4.18 berikut ini:

Tabel 4.18 Tabel hasil pengenalan wajah penambahan kelas kedua

SVM Update 2		Hasil Pencarian		Jml	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Wajah 1	4	2	2	4	1.390	50%	50%	91%
Wajah 2	4	1	2	3	1.293	33%	25%	89%
Wajah 3	4	1	1	2	1.577	50%	25%	91%
Wajah 4	4	0	1	1	1.356	0%	0%	89%
Wajah 5	4	1	0	1	1.354	100%	25%	93%
Wajah 6	4	2	1	3	1.392	67%	50%	93%
Wajah 7	4	1	1	2	1.331	50%	25%	91%
Wajah 8	4	1	0	1	1.476	100%	25%	93%
Wajah 9	4	3	7	10	1.753	30%	75%	82%
Wajah 10	4	2	5	7	1.799	29%	50%	84%
Wajah 11	3	2	8	10	1.127	20%	67%	80%
Wajah 12	2	0	1	1	1.533	0%	0%	93%
Rata-rata					1.448	44%	35%	89%
W. Train	6.969							

Pada penambahan kedua ini, sistem menunjukkan performa yang tidak jauh berbeda dengan performa pada sistem sebelumnya, dimana *precision* mengalami peningkatan 1% namun diikuti dengan penurunan nilai *recall* sebesar 2% dengan nilai akurasi tetap di nilai 89%. Namun pada penambahan yang kedua ini, sistem tidak mampu mengenali 2 data baru yang ditambahkan pada sistem dengan benar. Untuk waktu selama proses *training*, dengan bertambahnya satu kali proses *training* dari sistem sebelumnya, waktu komputasi selama proses *training* bertambah sebesar 0.501 detik.

Selanjutnya, dilakukan penambahan satu kelas lagi berupa citra wajah dengan tipe baru sebanyak 5 citra, dengan 4 citra untuk proses *training* dan 1 citra untuk proses *testing*. Hasilnya bisa dilihat pada Tabel 4.19 berikut ini:

Tabel 4.19 Tabel hasil pengenalan wajah penambahan kelas ketiga

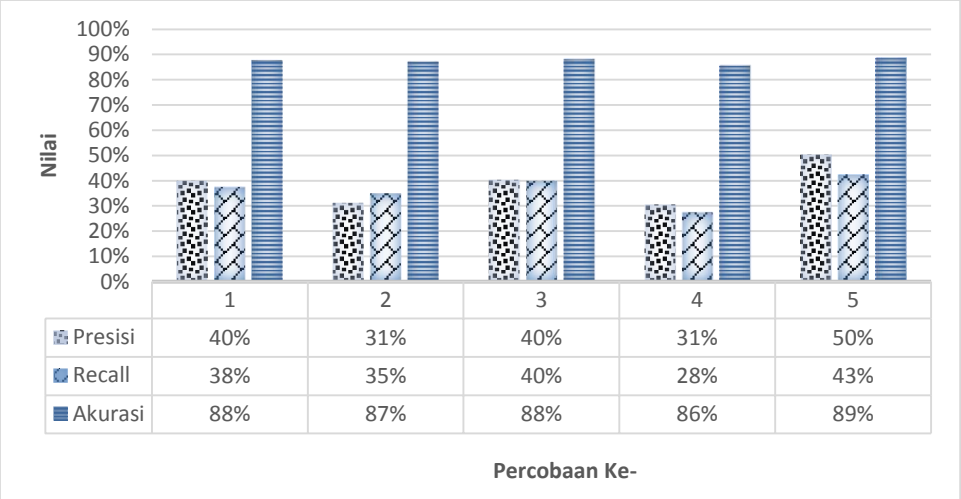
SVM Update 3		Hasil Pencarian		JMl	Hasil			
Jml Sebenarnya		Benar	Salah		Waktu *Testing*	*Precision*	*Recall*	Akurasi
Wajah 1	4	2	2	4	1.377	50%	50%	91%
Wajah 2	4	1	2	3	1.442	33%	25%	89%
Wajah 3	4	1	1	2	1.450	50%	25%	91%
Wajah 4	4	0	1	1	1.416	0%	0%	89%
Wajah 5	4	1	0	1	1.366	100%	25%	93%
Wajah 6	4	1	1	2	1.580	50%	25%	91%
Wajah 7	4	1	1	2	1.362	50%	25%	91%
Wajah 8	4	1	0	1	1.366	100%	25%	93%
Wajah 9	4	2	6	8	1.510	25%	50%	83%
Wajah 10	4	2	4	6	1.689	33%	50%	87%
Wajah 11	3	2	8	10	1.550	20%	67%	80%
Wajah 12	2	1	3	4	1.730	25%	50%	91%
Wajah 13	1	0	2	2	1.474	0%	0%	93%
Rata-rata					1.485	41%	32%	90%
Waktu *Training*	7.616							

Pada penambahan ketiga ini, dilihat dari hasil yang ditunjukkan oleh Tabel 4.19, terlihat bahwa performa sistem tidak banyak mengelami perubahan dari sistem sebelumnya, nilai *precision* dan *recall* sama-sama hanya mengalami penurunan nilai sebesar 3% sedangkan nilai akurasi meningkat 1% menjadi 90%. Waktu komputasi selama proses *training* pun bertambah 0.647 detik untuk satu kali proses yang lebih banyak dari sebelumnya.

Dari hasil klasifikasi pengenalan wajah, nilai akurasi yang didapatkan oleh sistem mencapai 89% sedangkan selama tiga kali penambahan kelas, nilai akurasi yang dicapai cenderung sama yaitu 89%, 89% dan 90%. Sedangkan untuk nilai *precision* dibandingkan dengan kelas awal yaitu 38%, nilainya cenderung tidak jauh berbeda selama penambahan kelas yaitu 43%, 44% dan 41%. Untuk nilai *recall* dimana yang semula nilainya 43% mengalami penurunan menjadi 37%, 35% dan 32% untuk tiga kali penambahan kelas secara berturu-turut.

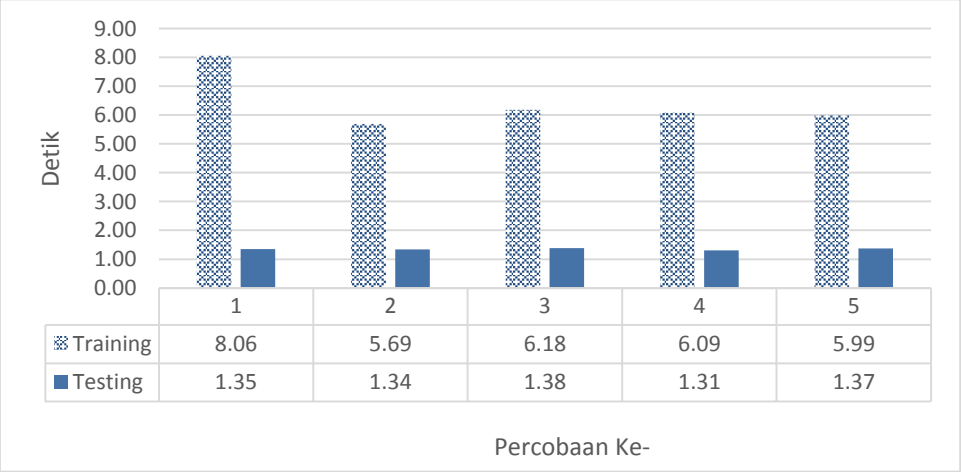
Sedangkan jika dilihat dari segi waktu komputasinya, hasil dari klasifikasi pengenalan wajah menunjukkan waktu pembelajarannya tidak bertambah cepat namun jika diperhatikan bahwa data pembelajaran yang digunakan untuk proses klasifikasi wajah jauh lebih sedikit dari proses klasifikasi objek umum, sehingga bisa dikatakan reduksi data dari data asli menjadi *support vector* pun tidaklah besar, akibatnya perbedaan waktu komputasi selama pembelajaran yang bertambah ini tidak terlihat.

Grafik dari hasil klasifikasi pengenalan wajah untuk seluruh percobaan yang dilakukan disajikan pada Gambar 4.16 sampai dengan Gambar 4.23:



	1	2	3	4	5
Presisi	40%	31%	40%	31%	50%
Recall	38%	35%	40%	28%	43%
Akurasi	88%	87%	88%	86%	89%

Gambar 4.16. Nilai *precision*, *recall*, akurasi 10 kelas pengenalan wajah

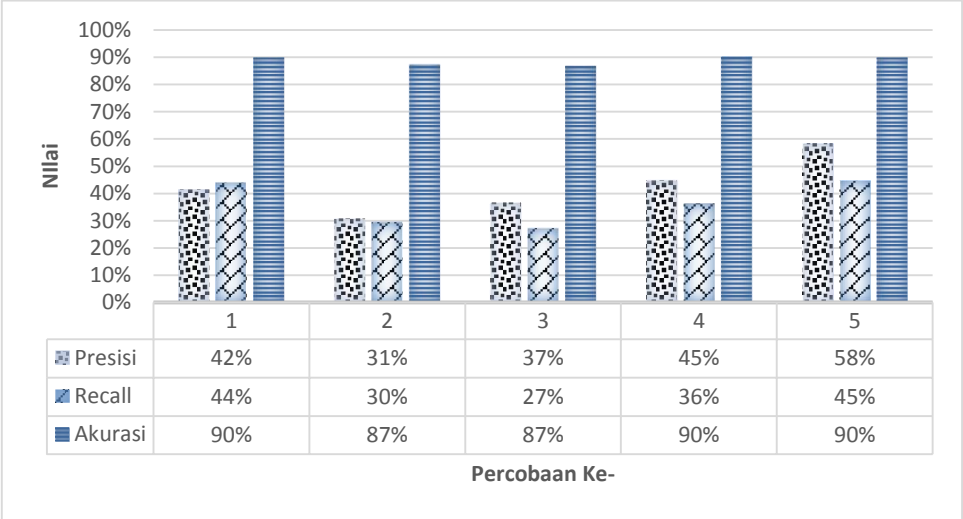


	1	2	3	4	5
Training	8.06	5.69	6.18	6.09	5.99
Testing	1.35	1.34	1.38	1.31	1.37

Gambar 4.17. Waktu komputasi 10 kelas pengenalan wajah

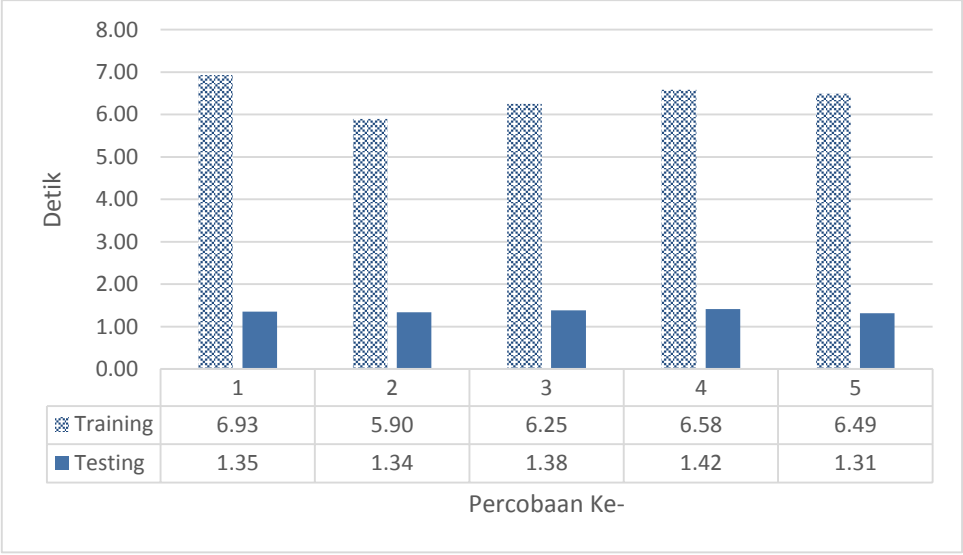
Gambar 4.16 dan 4.17 menunjukkan performa dan waktu komputasi dari sistem yang mengklasifikasikan 10 wajah dengan metode SVM berbasis multi kernel, dari 5 percobaan yang dilakukan secara *cross validation*, sama halnya dengan percobaan sebelumnya, pada klasifikasi wajah ini sistem juga menunjukkan grafik performa yang cukup konstan artinya tidak ada perbedaan nilai yang besar antara percobaan pertama, kedua, hingga terakhir.

Selanjutnya, Gambar 4.18 dan Gambar 4.19 berikut ini menunjukkan performa dari sistem saat dilakukan penambahan satu kelas pada sistem sebelumnya.



	1	2	3	4	5
Presisi	42%	31%	37%	45%	58%
Recall	44%	30%	27%	36%	45%
Akurasi	90%	87%	87%	90%	90%

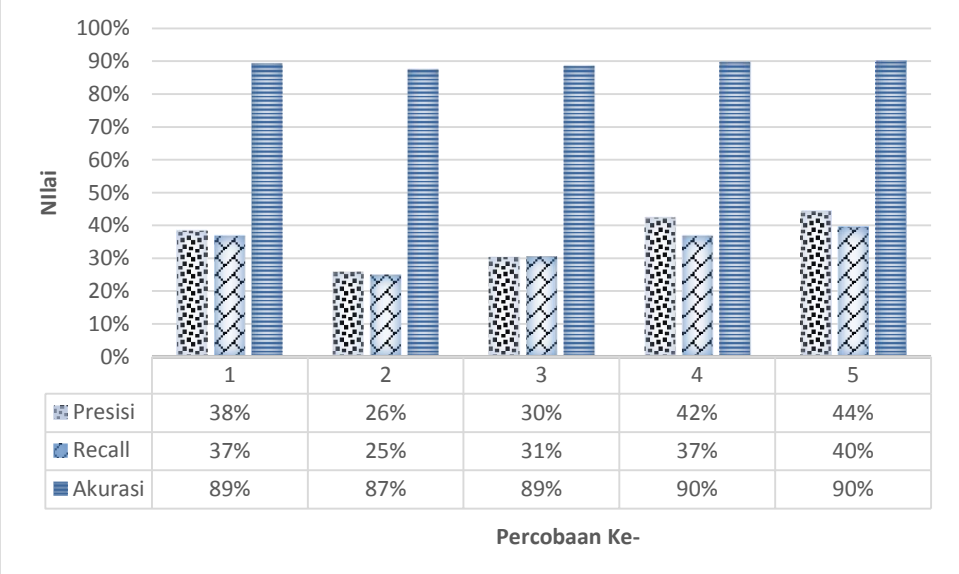
Gambar 4.18. Nilai *precision*, *recall*, akurasi pengenalan wajah penambahan kelas pertama



	1	2	3	4	5
Training	6.93	5.90	6.25	6.58	6.49
Testing	1.35	1.34	1.38	1.42	1.31

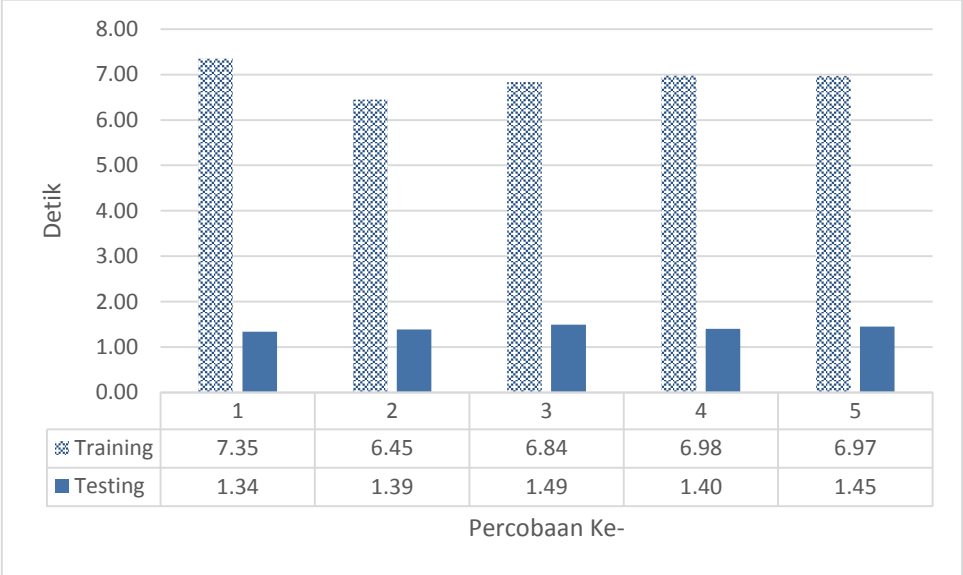
Gambar 4.19. Waktu komputasi pengenalan wajah penambahan kelas pertama

Seperti halnya percobaan pada citra dengan objek umum, pada penambahan kelas yang pertama ini secara umum pola dari nilai performa dan waktu komputasi pada percobaan pertama sampai terakhir cenderung sama dan konstan seperti pada percobaan sebelum ditambahkan satu kelas baru pada sistem, namun jika diperhatikan antara Gambar 4.16 dan 4.18 terjadi anomali pada percobaan keempat, dimana pada percobaan pertama atau klasifikasi sistem dengan lima kelas nilai performa dari percobaan keempat lebih tinggi dari percobaan ketiga namun pada sistem kedua atau penambahan satu kelas setelahnya, performa dari percobaan keempat lebih tinggi dari percobaan ketiga, hal ini bisa saja terjadi karena dengan data yang terbilang kecil titik-titik *support vector* yang menentukan garis pemisahpun terbilang kecil, sehingga jika terjadi sedikit saja perubahan pada beberapa *support vector* pada pembelajaran yang baru, maka garis pemisahpun juga berubah, hal ini berbeda jika data pembelajaran yang terlibat jauh lebih besar, maka kemungkinan perubahan garis keputusan akibat beberapa perubahan *support vector* pada pembelajaran yang baru semakin kecil.



	1	2	3	4	5
Presisi	38%	26%	30%	42%	44%
Recall	37%	25%	31%	37%	40%
Akurasi	89%	87%	89%	90%	90%

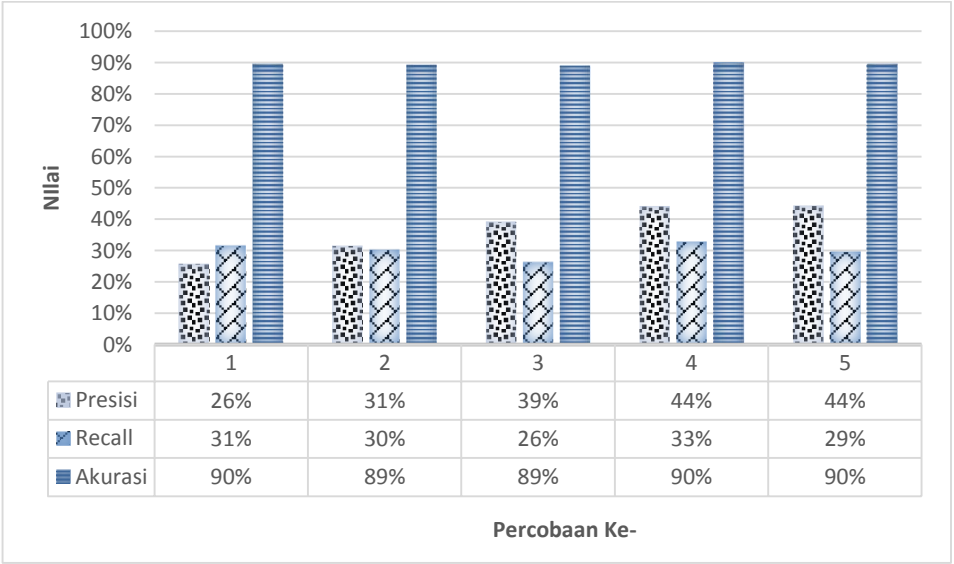
Gambar 4.20. Nilai *precision*, *recall*, akurasi pengenalan wajah penambahan kelas kedua



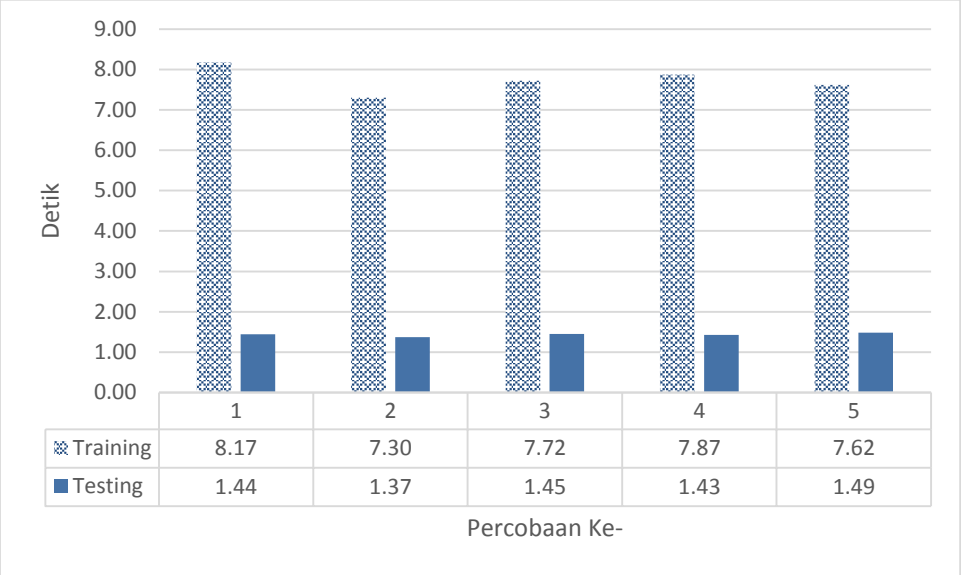
Gambar 4.21. Waktu komputasi pengenalan wajah penambahan kelas kedua

Pada grafik yang ditunjukkan oleh Gambar 4.20 dan 4.21, dapat dilihat bahwa pada penambahan satu kelas yang kedua kalinya pada sistem menunjukkan bahwa sistem masih memiliki pola performa yang tidak jauh berbeda dengan penambahan yang pertama.

Untuk performa sistem dengan penambahan kelas yang ketiga ditunjukkan pada Gambar 4.22 dan 4.23 berikut:



Gambar 4.22. Nilai *precision*, *recall*, akurasi pengenalan wajah penambahan kelas ketiga



Gambar 4.23. Waktu komputasi pengenalan wajah penambahan kelas ketiga

Pada penambahan yang ketiga kalinya ini, sama seperti pada kasus sebelumnya yang terjadi pada sistem awal (sebelum penambahan) dengan sistem penambahan satu kelas yang pertama, pola performa yang ditunjukkan pada Gambar 4.22 juga menunjukkan perbedaan dengan 4.20 atau pola performa pada sistem sebelumnya, seperti yang terlihat bahwa pada sistem sebelumnya atau sistem dengan penambahan kelas yang kedua nilai *precision* dan *recall* pada percobaan pertama lebih baik nilainya dari percobaan kedua, namun pada sistem penambahan ketiga yang ditunjukkan Gambar 4.22 nilai *precision* dan *recall* pada percobaan pertama tidak lebih baik nilainya dari percobaan kedua.

Berdasarkan grafik yang ditunjukkan pada Gambar 4.8 sampai Gambar 4.23 dapat disimpulkan bahwa secara empiris berdasarkan percobaan yang telah dilakukan, metode klasifikasi SVM berbasis multi kernel dengan pembelajaran yang bertambah memiliki performa yang konstan dari rata-rata keseluruhan percobaan yang dilakukan secara *cross validation.* Selain itu, dari grafik tersebut dapat juga dilihat bahwa antara sistem pada SVM multi kernel dengan SVM multi kernel dengan pembelajaran yang bertambah dari yang pertama sampai ketiga memiliki pola hasil performa yang cenderung sama. Hal ini berarti, informasi yang didapatkan pada hasil *training* di awal (klasifikasi 5 kelas) tidak banyak berubah meskipun dilakukan pembelajaran baru tanpa harus menggunakan keseluruhan dari

data *training* yang digunakan sebelumnya. Hanya saja, beberapa kali terjadi anomali pada percobaan pengenalan wajah. Hal ini dikarenakan sedikitnya data *training* yang digunakan sehingga *support vector* yang menentukan garis pemisah juga berjumlah sedikit. Akibatnya, jika terjadi perubahan sedikit saja pada data *support vector*, maka hal tersebut berpengaruh secara signifikan pula pada garis pemisah antar kelas.

4.5.3 Perbandingan Hasil Ujicoba

Untuk mengetahui performa dari metode SVM yang telah dimodifikasi, metode ini dibandingkan dengan metode SVM yang standar. Oleh karena itu, pada bagian berikutnya disajikan perbandingan performa rata-rata dari metode SVM multi kernel yang bertambah dengan metode SVM standar dengan satu kernel yaitu kernel polynomial (2.9) dengan order 3 dan kernel RBF (2.10). Berikut adalah hasil perbandingannya (Tabel lengkap hasil percobaan SVM Kernel Polynomial dan RBF bisa dilihat pada Lampiran A):

Tabel 4.20 Tabel perbandingan SVM multi kernel pada penambahan kelas pertama dengan SVM standar 6 kelas.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Pesawat	**58%**	**44%**	**84%**	42%	19%	80%	0%	0%	82%
Motor	**65%**	66%	**87%**	47%	**71%**	81%	0%	0%	82%
Kucing	26%	**18%**	76%	**28%**	18%	77%	0%	0%	**82%**
Mobil	**41%**	**34%**	79%	37%	30%	78%	0%	0%	**82%**
Wajah	**55%**	54%	**84%**	47%	56%	80%	18%	**100%**	18%
Bunga	28%	**68%**	81%	**30%**	55%	84%	0%	0%	**91%**
Rata-Rata	**45%**	**47%**	**82%**	38%	41%	80%	3%	17%	73%

Keterangan: P = *Precision*, R = *Recall*, A = Akurasi

Pada Tabel 4.20 terlihat bahwa secara rata-rata nilai performa baik *precision*, *recall* maupun akurasi yang dihasilkan dari metode SVM multi kernel dengan

pembelajaran yang bertambah ini lebih baik daripada kedua metode SVM lainnya, terlebih dengan SVM dengan kernel RBF. Pembelajaran yang dilakukan dengan metode ini menghasilkan klasifikasi dimana semua data *testing* diklasifikasikan dalam satu kelas saja yaitu kelas wajah (lihat Tabel 4.21). Ini menunjukkan bahwa pembelajaran dengan kernel RBF ini tidak mampu mengklasifikasikan citra dengan baik terlebih pada sistem temu kembali citra yang dikerjakan dalam penelitian ini.

Tabel 4.21 *Confusion Matriks* hasil klasifikasi kelompok citra wajah pada SVM RBF

		Nilai Prediksi	
		True	***False***
Nilai Sebenarnya	***True***	50	0
	False	225	275

Tabel 4.21 menunjukkan hasil dari percobaan temu kembali citra dengan SVM RBF pada kelas wajah (Tabel lengkap hasil percobaan bisa dilihat pada Lampiran A), dari hasil terlihat bahwa sistem mengenali seluruh data testing sebagai data positif sedangkan sebenarnya jumlah dari data positif hanya 50. Hal inilah yang membuat nilai akurasi dan *precision* menjadi sangat kecil sedangkan nilai *recall* sangat besar, jika dilakukan perhitungan *precision*, *recall* dan akurasi berdasarkan Persamaan (3.2), (3.3) dan (3.4) dari Tabel 4.21 tersebut, maka akan didapatkan hasil sebagai berikut:

$$Precision = \frac{50}{50 + 225} = 0.181818 \times 100\% = 18\%$$

$$Recall = \frac{50}{50+0} = 1 \times 100\% = 100\%$$

$$\text{Akurasi} = \frac{50 + 0}{275} = 0.181818 \times 100\% = 18\%$$

Tabel 4.22 Tabel perbandingan SVM multi kernel pada penambahan kelas kedua dengan SVM standar 7 kelas.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Pesawat	**51%**	**44%**	**83%**	42%	28%	81%	0%	0%	83%
Motor	64%	68%	**88%**	**57%**	**70%**	86%	0%	0%	83%
Kucing	**31%**	20%	79%	24%	**22%**	75%	0%	0%	**83%**
Mobil	39%	34%	79%	35%	**34%**	78%	0%	0%	**83%**
Wajah	**54%**	52%	**84%**	51%	62%	83%	17%	**100 %**	17%
Bunga	27%	**64%**	82%	**30%**	47%	**86%**	0%	0%	91%
Burung	**36%**	**27%**	94%	34%	12%	**95%**	0%	0%	95%
Rata-Rata	**43%**	**44%**	**84%**	39%	39%	83%	2%	14%	76%

Keterangan: P = *Precision*, R = *Recall*, A = Akurasi

Pada perbandingan ini, terlihat pada Tabel 4.22 bahwa performa dari pembelajaran SVM berbasis multi kernel yang bertambah masih lebih baik jika dibandingkan dengan kernel Polynomial atau RBF dimana metode yang dikerjakan memiliki nilai *precision* 4% lebih besar dari SVM Polynomial dan 41% lebih besar dari SVM RBF, sedangkan nilai *recall* dari metode yang dikerjakan 4 % lebih besar dari nilai *recall* pada SVM Polynomial dan 30% lebih besar dari SVM RBF serta nilai akurasi yang lebih baik dengan nilai 84% daripada nilai akurasi pada SVM Polynomial yang hanya 83% dan SVM RBF yang hanya 76%.

Selanjutnya, Tabel 4.23 berikut adalah perbandingan pada saat ditambahkan satu lagi kelas baru pada metode SVM yang dikerjakan.

Tabel 4.23 Tabel perbandingan SVM multi kernel pada penambahan kelas ketiga dengan SVM standar 8 kelas.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Pesawat	**49%**	**40%**	83%	43%	28%	82%	0%	0%	83%
Motor	55%	54%	85%	**57%**	**65%**	**86%**	0%	0%	83%
Kucing	23%	14%	**78%**	**23%**	**23%**	74%	0%	0%	83%
Mobil	36%	26%	80%	38%	**34%**	80%	0%	0%	**83%**
Wajah	**48%**	54%	**83%**	46%	**62%**	81%	3%	20%	70%
Bunga	27%	64%	83%	**29%**	50%	**86%**	7%	**80%**	25%
Burung	**33%**	**20%**	94%	18%	9%	**95%**	0%	0%	95%
Kupu-kupu	5%	**10%**	91%	**22%**	8%	**96%**	0%	0%	97%
Rata-Rata	**35%**	**35%**	**85%**	**35%**	**35%**	**85%**	1%	13%	78%

Keterangan: P = *Precision*, R = *Recall*, A = Akurasi

Pada penambahan kelas yang ketiga ini, performa dari ketiga metode yang ditunjukkan pada Tabel 4.23 menunjukkan bahwa pada bagian ini SVM Polynomial dengan SVM Multi kernel yang bertambah memiliki nilai performa yang sama, hasil ini wajar terjadi karena seperti yang diketahui sebelumnya bahwa seiring dengan penambahan kelas maka terjadi sedikit penurunan performa pada sistem, sedangkan performa pada pembelajaran SVM standar, hanya dipengaruhi oleh kemampuan kernel serta karakteristik dari data klasifikasi yang digunakan bukan dari berapa jumlah kelas atau berapa kali jumlah *training* yang dilakukan selama proses, karena setiap proses *training*, data yang digunakan adalah keseluruhan dari data *training*. Sedangkan bila dibandingkan dengan SVM RBF metode ini masih jauh lebih baik, dimana pada SVM RBF data *testing* mengumpul hanya pada dua kelas.

Untuk hasil dari perbandingan klasifikasi pengenalan wajah, ditunjukkan pada Tabel 4.24, 4.25 dan 4.26 berikut ini:

Tabel 4.24 Tabel perbandingan SVM multi kernel pada penambahan kelas pertama dengan SVM standar 11 kelas pada pengenalan wajah.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Wajah 1	**67%**	50%	93%	73%	**55%**	**95%**	0%	0%	91%
Wajah 2	**33%**	25%	88%	28%	**30%**	84%	0%	0%	**91%**
Wajah 3	50%	**25%**	91%	50%	15%	92%	0%	0%	91%
Wajah 4	**50%**	**25%**	91%	23%	15%	89%	0%	0%	91%
Wajah 5	**50%**	**25%**	91%	20%	10%	89%	0%	0%	91%
Wajah 6	**67%**	**50%**	**93%**	23%	20%	89%	0%	0%	91%
Wajah 7	33%	25%	88%	**57%**	25%	90%	0%	0%	**91%**
Wajah 8	50%	25%	91%	**87%**	25%	**92%**	0%	0%	91%
Wajah 9	30%	75%	81%	**31%**	**80%**	82%	0%	0%	**91%**
Wajah 10	**22%**	50%	**79%**	19%	60%	71%	9%	**100 %**	9%
Wajah 11	25%	**33%**	88%	**33%**	20%	92%	0%	0%	**93%**
Rata-rata	**43%**	**37%**	**89%**	40%	32%	88%	1%	9%	84%

Keterangan: P = *Precision*, R = *Recall*, A = Akurasi

Pada Tabel 4.24 dapat dilihat bahwa untuk percobaan klasifikasi wajah, metode yang dikerjakan juga memiliki nilai performa yang lebih baik dari metode SVM Polynomial ataupun RBF, dan pada percobaan kali ini pun SVM RBF tidak mampu mengklasifikasikan citra wajah dengan baik, terbukti dengan hasil yang diperlihatkan bahwa semua data *testing* mengumpul pada satu kelas yaitu kelas pada wajah 10.

Berikutnya, pada Tabel 4.25 ditunjukkan hasil perbandingan sistem yang mengalami penambahan satu kelas baru dengan sistem dengan pembelajaran SVM Polynomial dan RBF standar.

Tabel 4.25 Tabel perbandingan SVM multi kernel pada penambahan kelas kedua dengan SVM standar 12 kelas pada pengenalan wajah.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Wajah 1	50%	50%	91%	**73%**	**55%**	**95%**	0%	0%	91%
Wajah 2	**33%**	25%	89%	26%	**30%**	84%	0%	0%	**91%**
Wajah 3	50%	**25%**	91%	**70%**	20%	92%	0%	0%	91%
Wajah 4	0%	0%	89%	**13%**	**10%**	89%	0%	0%	**91%**
Wajah 5	**100%**	25%	93%	90%	25%	**93%**	0%	0%	91%
Wajah 6	**67%**	**50%**	**93%**	40%	20%	91%	0%	0%	91%
Wajah 7	50%	**25%**	91%	**53%**	20%	90%	0%	0%	91%
Wajah 8	40%	**20%**	92%	**47%**	15%	92%	0%	0%	91%
Wajah 9	**100%**	25%	**93%**	34%	**75%**	83%	0%	0%	91%
Wajah 10	**30%**	75%	82%	25%	**80%**	75%	0%	0%	**91%**
Wajah 11	**29%**	67%	80%	25%	33%	**88%**	7%	**100 %**	8%
Wajah 12	0%	0%	93%	40%	20%	96%	40%	20%	96%
Rata-rata	44%	**35%**	89%	**45%**	34%	89%	4%	10%	85%

Keterangan: P = *Precision,* R = *Recall,* A = Akurasi

Pada Tabel 4.25 terlihat bahwa secara rata-rata nilai akurasi dari SVM yang dikerjakan dengan SVM Polynomial memiliki nilai yang sama, sedangkan untuk nilai *precision,* hasil terbaik ditunjukkan oleh SVM polynomial dengan selisih 1% bila dibandingkan dengan metode yang dikerjakan namun untuk nilai *recall,* metode yang dikerjakan unggul 1% lebih besar dari SVM Polynomial. Sedangkan SVM RBF masih menunjukkan performa yang buruk, dimana kali ini hasil klasifikasi data *testing* hanya menghasilkan dua kelas saja.

Berikutnya dibandingkan hasil klasifikasi SVM Polynomial dan RBF pada 13 tipe wajah berbeda dengan metode pembelajaran yang dikerjakan (hasil dari penambahan satu kelas baru pada sistem sebelumnya)

Tabel 4.26 Tabel perbandingan SVM multi kernel pada penambahan kelas ketiga dengan SVM standar 13 kelas pada pengenalan wajah.

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
Wajah 1	50%	50%	91%	**73%**	**55%**	**95%**	0%	0%	91%
Wajah 2	**33%**	25%	89%	30%	**30%**	86%	0%	0%	**91%**
Wajah 3	50%	**25%**	91%	**70%**	20%	**93%**	0%	0%	91%
Wajah 4	0%	0%	89%	**23%**	**15%**	89%	0%	0%	**91%**
Wajah 5	**100%**	25%	**93%**	70%	25%	92%	0%	0%	91%
Wajah 6	**50%**	**25%**	91%	40%	20%	91%	0%	0%	91%
Wajah 7	50%	25%	91%	**63%**	25%	**92%**	0%	0%	91%
Wajah 8	**100%**	**25%**	**93%**	47%	15%	91%	0%	0%	91%
Wajah 9	25%	50%	83%	**35%**	**60%**	87%	0%	0%	**91%**
Wajah 10	**33%**	50%	83%	26%	**65%**	81%	0%	0%	**91%**
Wajah 11	**20%**	67%	80%	16%	27%	**84%**	7%	**100%**	7%
Wajah 12	25%	**50%**	91%	24%	30%	90%	**40%**	20%	**97%**
Wajah 13	0%	0%	93%	0%	0%	93%	0%	0%	**98%**
Rata-rata	**41%**	**32%**	**90%**	40%	30%	89%	4%	9%	86%

Keterangan: P = *Precision*, R = *Recall*, A = Akurasi

Hasil percobaan pada penambahan kelas yang ketiga ini, yang ditunjukkan oleh Tabel 4.26 menunjukkan performa yang tidak berbeda jauh dari sistem sebelumnya, namun kali ini performa terbaik ditunjukkan oleh metode yang dikerjakan, karena pada kali ini nilai performa dari SVM Polynomial menurun dari nilai sebelumnya. Sedangkan pada SVM RBF, hasil klasifikasinya tidak berbeda dengan hasil pada klasifikasi 12 kelas dimana hasilnya hanya mengelompok pada 2 kelas saja yaitu wajah 11 dan wajah 12.

Berdasarkan hasil yang ditunjukkan pada tabel perbandingan tersebut, terlihat bahwa secara empiris, performa rata-rata baik dari *precision*, *recall* maupun akurasi, metode SVM multi kernel dengan pembelajaran yang bertambah ini memiliki nilai yang baik, bahkan seringkali lebih baik dari SVM dengan kernel lainnya khususnya RBF dan Polynomial, jika diambil rata-rata dari semua

percobaan, baik dalam klasifikasi objek umum maupun pengenalan wajah maka didapatkan *precision* dari SVM yang dikerjakan 2% lebih baik daripada SVM Polynomial dan 39% daripada SVM RBF, kemudian nilai *recall* 3% dan 26% lebih baik daripada SVM Polynomial maupun SVM RBF, dari sisi akurasipun menunjukkan hasil yang sama, yaitu metode SVM yang dikerjakan memiliki nilai 1% lebih baik bila dibandingkan dengan SVM Polynomial dan 7% lebih baik daripada SVM RBF (Lihat Tabel 4.27).

Hasil percobaan juga menunjukkan bahwa SVM RBF memiliki nilai *precision* dan *recall* yang buruk jika diaplikasikan pada temu kembali citra seperti yang diuji cobakan dalam penelitian ini, ini berarti SVM dengan kernel RBF tidak sesuai jika digunakan untuk mengklasifikasikan data-data citra seperti diatas. Hal ini mendukung pernyataan sebelumnya pada bagian latar belakang dari penelitian ini bahwa penentuan kernel yang tepat merupakan hal yang penting dalam menyelesaikan masalah klasifikasi dengan metode SVM, karena tidak semua kernel baik digunakan untuk semua masalah klasifikasi.

Tabel 4.27 Tabel rata-rata perbandingan performa

	SVM yang dikerjakan			SVM Polynomial			SVM RBF		
	P	R	A	P	R	A	P	R	A
A	**45%**	**47%**	**82%**	38%	41%	80%	3%	17%	73%
B	**43%**	**44%**	**84%**	39%	39%	83%	2%	14%	76%
C	35%	35%	85%	35%	35%	85%	1%	13%	78%
D	**43%**	**37%**	**89%**	40%	32%	88%	1%	9%	84%
E	44%	**35%**	89%	**45%**	34%	89%	4%	10%	85%
F	**41%**	**32%**	**90%**	40%	30%	89%	4%	9%	86%
Rata-rata	**42%**	**38%**	**87%**	40%	35%	86%	3%	12%	80%

Keterangan:

P = *Precision*, R = *Recall*, A = Akurasi

A = Klasifikasi objek umum SVM penambahan kelas pertama atau SVM 6 kelas RBF/Polynomial

B = Klasifikasi objek umum SVM penambahan kelas kedua atau SVM 7 kelas RBF/Polynomial

C = Klasifikasi objek umum SVM penambahan kelas ketiga atau SVM 8 kelas RBF/Polynomial

D = Klasifikasi wajah SVM penambahan kelas pertama atau SVM 11 kelas RBF/Polynomial
E = Klasifikasi wajah SVM penambahan kelas kedua atau SVM 12 kelas RBF/Polynomial
F = Klasifikasi wajah SVM penambahan kelas ketiga atau SVM 13 kelas RBF/Polynomial

Tabel 4.28 berikut ini adalah tabel perbandingan dari waktu komputasinya, dimana perhitungan waktu *testing* diambil dari rata-rata perhitungan waktu *testing* untuk seluruh kategori pada tiap percobaan:

Tabel 4.28 Tabel perbandingan waktu komputasi

	SVM yang dikerjakan		SVM Polynomial		SVM RBF	
	Training	*Testing*	*Training*	*Testing*	*Training*	*Testing*
A	**8.1268**	**11.061**	9.0066	12.911	10.935	27.557
B	**8.9122**	**12.265**	10.8303	12.653	13.138	25.406
C	**9.1613**	**12.094**	12.5897	13.305	14.291	24.848
Rata-rata	**8.7334**	**11.8067**	10.8089	12.9563	12.7880	25.9370

Keterangan:

A = Perbandingan SVM penambahan kelas pertama dengan SVM 6 kelas
B = Perbandingan SVM penambahan kelas kedua dengan SVM 7 kelas
C = Perbandingan SVM penambahan kelas ketiga dengan SVM 8 kelas

Pada Tabel 4.28 terlihat bahwa pembelajaran SVM yang bertambah membutuhkan waktu komputasi yang lebih cepat daripada metode SVM standar dengan jumlah kelas yang sama serta jumlah data yang sama pula, hal ini sesuai dengan apa yang dijelaskan sebelumnya bahwa pada pembelajaran yang bertambah ini data yang digunakan untuk proses *training* merupakan data *support vector* yang mana jumlahnya tentu lebih kecil daripada jumlah awal dari data pembelajaran, sehingga waktu komputasinya pun semakin cepat.

Untuk proses *testing* perbedaan waktu komputasi tidak terlalu berbeda, hal ini dikarenakan proses *testing* hanya bergantung pada jumlah data *testing* dan jumlah model pembelajaran dimana dalam percobaan ini semua metode melakukan hal

yang sama dengan jumlah model dan data *testing* yang sama. Untuk kasus SVM RBF terlihat bahwa waktu *testing* jauh lebih lama dari metode lainnya, hal ini hanya dikarenakan pada proses penampilan citra hasil pencarian pada *user interface* temu kembali citra yang melambat karena jumlah *figure* yang ditampilkan terlalu banyak dalam sekali *testing*, karena seperti diketahui sebelumnya bahwa pada SVM ini hasil klasifikasinya hanya mengumpul pada satu atau dua kelas saja.

Selanjutnya, Tabel 4.29 adalah tabel yang menunjukkan hasil perbandingan waktu komputasi untuk percobaan klasifikasi pengenalan wajah, berikut adalah hasilnya:

Tabel 4.29 Tabel perbandingan waktu komputasi pengenalan wajah

	SVM yang dikerjakan		SVM Polynomial		SVM RBF	
	Training	*Testing*	*Training*	*Testing*	*Training*	*Testing*
A	**6.4292**	1.3603	6.4952	1.2080	6.6230	0.9197
B	**6.9161**	1.4147	7.2064	1.2115	7.3084	0.9586
C	7.7361	1.4342	**7.4446**	1.2354	8.0823	1.1031
Rata-rata	**7.0271**	1.4031	7.0487	1.2183	7.3379	**0.9938**

Keterangan:

A = Perbandingan SVM penambahan kelas pertama dengan SVM 11 kelas

B = Perbandingan SVM penambahan kelas kedua dengan SVM 12 kelas

C = Perbandingan SVM penambahan kelas ketiga dengan SVM 13 kelas

Pada Tabel 4.29 perbedaan waktu komputasi untuk proses *training* antara metode satu dengan lainnya tidaklah terlalu signifikan, meskipun untuk klasifikasi 11 kelas dan 12 kelas, metode yang dikerjakan memiliki waktu tercepat. Seperti yang dijelaskan sebelumnya bahwa adanya perbedaan yang tidak signifikan ini lebih dikarenakan pada jumlah data *training* yang tidak terlalu besar, sehingga reduksi dari jumlah data awal dengan data *support vector* yang digunakan pada proses *training* berikutnya pun tidak terlalu signifikan. Akibatnya pengaruh pada waktu komputasi selama *training* pun tidak terlihat secara signifikan pula.

BAB 5
KESIMPULAN DAN SARAN

Pada bab ini disampaikan kesimpulan dan saran yang diperoleh dari penerapan metode SVM multi kernel pada temu kembali citra dengan pembelajaran yang bertambah yang telah dijelaskan pada bab sebelumnya.

5.1 Kesimpulan

Dari hasil penelitian pada bab sebelumnya, diperoleh hasil sebagai berikut:

1. Metode klasifikasi SVM multi kernel dengan pembelajaran yang bertambah didapat dengan mengkombinasikan kernel-kernel Linear, Polynomial dan RBF dimana kemudian setelah *training* dilakukan maka didapatkan *support vector* yang digunakan dalam proses *training* SVM berikutnya. Sedangkan untuk mendapatkan klasifikasi multi kelas pada SVM yang merupakan model klasifikasi kelas biner maka pembelajaran dilakukan dengan pendekatan yang disebut dengan metode *one-against-all*. Untuk menerapkannya pada temu kembali citra, citra yang diklasifikasikan diproses dan didapatkan nilai histogramnya, nilai histogram inilah yang dihitung dan diklasifikasikan dengan metode SVM yang telah didapatkan sebelumnya.
2. Hasil yang diperoleh dari penerapan metode klasifikasi multi kernel dengan pembelajaran bertambah untuk temu kembali citra memperlihatkan tingkat akurasi yang lebih baik daripada metode klasifikasi dengan kernel tunggal Polynomial ataupun RBF dengan pembelajaran yang standar. Dari keseluruhan percobaan menunjukkan bahwa metode yang dikerjakan memiliki rata-rata akurasi sebesar 87% lebih baik dari SVM RBF yang hanya 80% ataupun SMV Polynomial yang sebesar 80%.

   Selain tingkat akurasi, metode yang dikerjakan juga memiliki nilai *precision* dan *recall* lebih baik daripada SVM dengan kernel tunggal Polynomial maupun RBF dengan nilai rata-rata *recall* mencapai 38% selisih 3% dari SVM Polynomial dan selisih 26% dari SVM RBF, sedangkan nilai *precision* yang didapat dari metode yang dikerjakan sebesar 42% lebih baik daripada SVM Polynomial yang bernilai 40% serta SVM RBF yang hanya 3% saja.

Hasil penelitian juga menunjukkan bahwa metode pembelajaran yang bertambah membutuhkan waktu yang lebih cepat saat proses *training* berlangsung, daripada dengan pembelajaran yang biasa. Untuk mengklasifikasikan 2750 data dengan 11 kelas, 12 kelas dengan 2900 data dan 13 kelas dengan 3000 data secara bertahap dibutuhkan waktu sebesar 8.1268, 8.9122 dan 9.1613 detik. Hal ini lebih cepat daripada dengan pembelajaran biasa untuk data dan kelas yang sama dimana metode ini membutuhkan waktu 9.0066, 10.8303 dan 12.5897 detik untuk kernel Polynomial dan 10.935, 13,138 dan 14,291 detik untuk kernel RBF.

5.2 Saran

Berdasarkan penelitian ini, saran untuk penelitian berikutnya adalah:

1. Metode pembelajaran multi kernel ini bisa dicoba dengan mengkombinasikan kernel-kernel lain selain dari 3 kernel yang telah digunakan dalam penelitian ini.
2. Uji coba bisa dilakukan dengan jumlah data yang jauh lebih besar dari yang digunakan dalam penelitian ini sehingga nanti bisa didapatkan perbedaan waktu komputasi yang lebih signifikan dari metode pembelajaran standar.
3. Untuk mendapatkan hasil terbaik, disarankan untuk menggunakan data citra dengan *background* yang homogen.
4. Penelitian selanjutnya dapat dilakukan dengan penerapan pada aplikasi lain atau jenis data lain. Misalnya untuk deteksi objek pada video atau pengenalan suara, atau dengan penerapan yang sama namun dengan ekstraksi fitur lainnya atau *pre-processing* yang berbeda, misalnya dengan segmentasi atau lainnya.

DAFTAR PUSTAKA

Abe, S. (2010), *Support Vector Machines for Pattern Classification*, Springer-Verlag, London, UK.

Athoillah, M., Irawan, M.I., dan Imah, E.M. (2015), "Study Comparison of SVM-, K-NN- and Backpropagation-Based Classifier for Image Retrieval", *Journal of Computer Science and Information,* Vol 8, No 1, hal-11-19

Campbell, C dan Ying, Y. (2011), *Learning with Support Vector Machines,* Buku seri *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool Publisher, UK.

Clarke, B., Fokoue, E., dan Zhang, H.H. (2009), *Principles and Theory for Data Mining and Machine Learning,* Springer Science + Bussiness Media, New York, USA.

Datta, R., Joshi, D., Li, J., dan Wang, J.Z., (2008), "Image Retrieval : Ideas, Influence and Trends of the New Age" *ACM Computing Survey*, Vol 40, No. 2, Article 5

Dinnes, K.R. dan Hariharan, S., (2014), "Approaches and Trends in Content Based Image Retrieval" *Proceesings of 24th International Conference in Communication, Network and Computing, CNC,* hal 473-477

Domeniconi, C dan Gunopulos, D., (2001), "Incremental Support Vector Machine Construction" *Proceesings of 1st IEEE International Conference on Data Mining (ICDM '01),* San Jose, USA. hal 641-642.

Gonen, M. dan Alpaydin, E. (2011), "Multiple Kernel Learning Algorithms", *Journal Machine Learning Research 12*, hal 2211-2268

Gosselin, P.H., Precioso, F. dan Philip-Foliguet, S. (2010), "Incremental Kernel Learning for Active Image Retrieval without Global Dictionaries", Jurnal *Pattern Recognition 44*, hal 2244-2254

Hamel, L. (2009), *Knowledge Discovery with Support Vector Machine Learning*, John Wiley & Sons, Inc, New Jersey, USA

Imah, E.M., Jatmiko, W., dan Basaruddin, T. (2012), "Adaptive Multilayer Generalized Learning Vector Quantization (AMGLVQ) as new algorithm with integrating feature extraction and classification for Arrhythmia heartbeats classification", *Systems, Man, and Cybernetics (SMC), IEEE International Conference*. Seoul, hal 150-155.

Kumar, A.R., dan Saravanan, D., (2013). "Content Based Image Retrieval Using Color Histogram", *International Journal of Computer Science dan Information Technologies (IJCSIT)*, Vol 4, No 2, Hal 242-245

Shinya, K dan Shigeo, A. (2006), *Incremental Training of Support Machine Learning using Truncated Hypercones*, Lecture Notes in Computer Science-Artificial Neural Network in Pattern Recognition, Kobe University Repository, Jepang. hal 153-164

Lei, B,. Tan, E., C, S., Ni, D., dan Wang, T., (2015). "Saliency-driven image classification method based on histogram mining and image score", *Jurnal Pattern Recognition*, Vol 48, No 8, Hal 2567-2580

Li, D., Wang, J., Zhao, X., Liu, Y., dan Wang, D. (2014). "Multiple Kernel-Based Multi-Instance Learning Algorithm for Image Classification", *Jurnal Visual Communication and Image Representation*, Vol 25, No 5, Hal 1112-1117

Liu, X., Zhou, L., Wang, L., Zhang, J., Yin, J., dan Shen, D., (2015). "An Efficient Radius-Incorporated MKL Algorithm for Alzheimare's Disease Prediction", *Jurnal Pattern Recognition*, Vol 48, No 7, Hal 2141-2150

Lusiyanti, D., dan Irawan, M.I. (2014), *Perbandingan Metode Learning Vector Quantization (LVQ) dan Support Vector Machine (SVM) untuk Prediksi Penyakit Jantung Koroner*. Thesis S2, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember, Surabaya.

Mohri, M., Rostamizadeh, A. dan Talwalkar, A. (2012), *Foundations of Machine Learning*, Buku seri *The Adaptive Computations and Machine Learning*, MIT Press, Massachusetts Institute of Technology, USA.

Murphy, K.P. (2012), *Machine learning A probabilistic Perspective*, Buku seri *The Adaptive Computations and Machine Learning*, MIT Press, Massachusetts Institute of Technology, USA.

Rakotomamonjy, A., Bach, F.R., Canu, S. Dan Grandvalet, Y. (2008), "SimpleMKL", *Journal Machine Learning Research 9*, hal 2491-2521

Rakotomamonjy, A., Bach, F.R., Canu, S. Dan Grandvalet, Y. (2007), "More Efficiency in Multiple Kernel Learning", *Proceesings of 24th International Conference on Machine Learning*, Corvallis, OR, USA. hal 775-782

Scholkopf, B dan Smola, A.j. (2001), *Learning with Kernels*, Buku seri *The Adaptive Computations and Machine Learning*, MIT Press, Massachusetts Institute of Technology, USA.

Shrivastava, A., Pillai, J.K., dan Patel, V.M. (2015). "Multiple Kernel-based Dictionary Learning for Weakly Supervised Classification", *Jurnal Pattern Recognition*, Vol 48, No 8, Hal 2667-2675

Steinwart, I dan Christmann, A. (2008), *Support Vector Machines,* Buku seri *Information Science and Statistics,* Springer Science + Bussiness Media, New York, USA.

Tomasouw, B.P., dan Irawan, M.I. (2012), *Multi-Class Twin Bounded SVM Berbasis Decision Directed Acyclic Graph (DDAG) untuk Pengenalan Suara*, Thesis S2, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Sepuluh Nopember, Surabaya.

Yeh, C.Y., Su, W.P., dan Lee, S.J., (2013). "An Efficient Multiple-Kernel Learning for Pattern Classification", *Jurnal Expert System with Application*, Vol 40, No 9, Hal 3491-3499

Zhang, Y., Hu, G., Zhu, F., dan Yu, J., (2009) "A new Incremental Learning Support Vector Machine", *International Converence on Artificial Intelligence and Computational Intelligence AICI*, Shanghai, hal 7-10

LAMPIRAN A

A.1 Hasil Klasifikasi objek umum SVM Polynomial

SVM 5 Kelas		Hasil Pencarian		JMl	Hasil			
Jml		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	10	11	20	11.491	48%	19%	80%
motor	50	37	34	71	14.367	52%	74%	81%
kucing	50	12	23	35	11.020	34%	24%	76%
mobil	50	24	34	58	12.551	42%	48%	76%
wajah	50	32	35	67	13.671	47%	63%	79%
Rata-rata					12.620	45%	46%	78%
W. Train	6.586							

SVM 6 Kelas		Hasil Pencarian		JMl	Hasil			
Jml		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	10	13	23	10.935	42%	19%	80%
motor	50	35	38	74	15.132	47%	71%	81%
kucing	50	9	22	31	11.927	28%	18%	77%
mobil	50	15	27	42	12.507	37%	30%	78%
wajah	50	28	32	60	14.283	47%	56%	80%
Bunga	25	14	32	46	12.684	30%	55%	84%
Rata-rata					12.911	38%	41%	80%
W. Train	9.007							

LAMPIRAN A (Lanjutan)

SVM 7 Kelas		Hasil Pencarian		JMl	Hasil			
Jml		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	14	18	32	12.100	42%	28%	81%
motor	50	35	26	61	13.836	57%	70%	86%
kucing	50	11	33	44	12.642	24%	22%	75%
mobil	50	17	31	48	13.094	35%	34%	78%
wajah	50	31	30	61	13.915	51%	62%	83%
Bunga	25	12	28	40	12.633	30%	47%	86%
Burung	15	2	2	3	10.355	34%	12%	95%
Rata-rata					12.653	39%	39%	83%
W. Train	10.830							

SVM 8 Kelas		Hasil Pencarian		JMl	Hasil			
Jml Sbnr		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	14	18	32	12.442	43%	28%	82%
motor	50	32	25	57	14.494	57%	65%	86%
kucing	50	11	38	49	14.125	23%	23%	74%
mobil	50	17	28	46	13.317	38%	34%	80%
wajah	50	31	36	67	15.525	46%	62%	81%
Bunga	25	12	30	43	13.704	29%	50%	86%
Burung	15	1	2	4	11.770	18%	9%	95%
Kupu-kupu	10	1	2	3	11.062	22%	8%	96%
Rata-rata					13.305	35%	35%	85%
W. Train	12.590							

LAMPIRAN A (Lanjutan)

A.2 Hasil Klasifikasi objek umum SVM RBF

SVM 5 Kelas		Hasil Pencarian		JMl	Hasil			
Jml Sbnr		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	0	0	0	8.321	0%	0%	80%
motor	50	0	0	0	8.643	0%	0%	80%
kucing	50	0	0	0	8.726	0%	0%	80%
mobil	50	40	160	200	84.159	16%	80%	32%
wajah	50	10	40	50	26.512	4%	20%	68%
Rata-rata					27.272	4%	20%	68%
W. Train	8.542							

SVM 6 Kelas		Hasil Pencarian		JMl	Hasil			
Jml Sbnr		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	0	0	0	10.118	0%	0%	82%
motor	50	0	0	0	10.526	0%	0%	82%
kucing	50	0	0	0	10.189	0%	0%	82%
mobil	50	0	0	0	10.444	0%	0%	82%
wajah	50	50	225	275	105.074	18%	100%	18%
Bunga	25	0	0	0	18.994	0%	0%	91%
Rata-rata					27.557	3%	17%	73%
W. Train	10.935							

LAMPIRAN A (Lanjutan)

SVM 7 Kelas		Hasil Pencarian		JMl	Hasil			
Jml Sbnr		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	0	0	0	10.923	0%	0%	83%
motor	50	0	0	0	10.900	0%	0%	83%
kucing	50	0	0	0	10.709	0%	0%	83%
mobil	50	0	0	0	10.346	0%	0%	83%
wajah	50	50	240	290	112.551	17%	100%	17%
Bunga	25	0	0	0	11.244	0%	0%	91%
Burung	15	0	0	0	11.171	0%	0%	95%
Rata-rata					25.406	2%	14%	76%
W. Train	13.138							

SVM 8 Kelas		Hasil Pencarian		JMl	Hasil			
Jml Sbnr		Benar	Salah		W.Testing	Precision	Recall	Akurasi
Pesawat	50	0	0	0	11.133	0%	0%	83%
motor	50	0	0	0	11.348	0%	0%	83%
kucing	50	0	0	0	10.918	0%	0%	83%
mobil	50	0	0	0	11.710	0%	0%	83%
wajah	50	10	50	60	32.237	3%	20%	70%
Bunga	25	20	219	239	98.080	7%	80%	25%
Burung	15	0	1	1	11.487	0%	0%	95%
Kupu2	10	0	0	0	11.874	0%	0%	97%
Rata-rata					24.848	1%	13%	78%
W. Train	14.291							

LAMPIRAN A (Lanjutan)

A.3 Hasil Klasifikasi wajah SVM Polynomial

SVM Kelas 10		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	2	0	3	1.160	73%	55%	95%
wajah 2	4	2	4	6	1.279	30%	40%	84%
wajah 3	4	0	0	1	0.845	30%	10%	91%
wajah 4	4	1	1	2	0.948	25%	20%	89%
wajah 5	4	1	1	1	0.867	30%	15%	90%
wajah 6	4	1	1	2	0.898	22%	15%	89%
wajah 7	4	1	1	2	0.910	27%	20%	90%
wajah 8	4	1	1	2	1.020	50%	20%	89%
wajah 9	4	3	8	11	1.543	29%	75%	78%
wajah 10	4	3	9	12	1.581	20%	70%	75%
Rata-rata					1.105	34%	34%	87%
W. Train	6.025							

SVM Kelas 11		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	2	0	3	1.154	73%	55%	95%
wajah 2	4	1	4	5	1.199	28%	30%	84%
wajah 3	4	1	0	1	0.953	50%	15%	92%
wajah 4	4	1	1	2	1.070	23%	15%	89%
wajah 5	4	0	1	1	1.136	20%	10%	89%
wajah 6	4	1	2	2	1.217	23%	20%	89%
wajah 7	4	1	1	2	1.138	57%	25%	90%
wajah 8	4	1	0	1	1.093	87%	25%	92%
wajah 9	4	3	7	10	1.495	31%	80%	82%
wajah 10	4	2	11	13	1.655	19%	60%	71%
wajah 11	3	1	1	2	1.179	33%	20%	92%
Rata-rata					1.208	40%	32%	88%
W. Train	6.495							

LAMPIRAN A (Lanjutan)

SVM Kelas 12		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	2	0	3	1.209	73%	55%	95%
wajah 2	4	1	4	5	1.390	26%	30%	84%
wajah 3	4	1	0	1	1.062	70%	20%	92%
wajah 4	4	0	1	2	1.083	13%	10%	89%
wajah 5	4	1	0	1	1.049	90%	25%	93%
wajah 6	4	1	1	2	1.094	40%	20%	91%
wajah 7	4	1	1	2	1.186	53%	20%	90%
wajah 8	4	1	0	1	0.873	47%	15%	92%
wajah 9	4	3	7	10	1.540	34%	75%	83%
wajah 10	4	3	10	14	1.734	25%	80%	75%
wajah 11	3	1	3	4	1.229	25%	33%	88%
wajah 12	2	0	0	1	1.090	40%	20%	96%
Rata-rata					1.212	45%	34%	89%
W. Train	7.206							

SVM Kelas 13		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	2	0	3	1.365	73%	55%	95%
wajah 2	4	1	4	5	1.416	30%	30%	86%
wajah 3	4	1	0	1	1.055	70%	20%	93%
wajah 4	4	1	2	2	1.217	23%	15%	89%
wajah 5	4	1	1	2	1.190	70%	25%	92%
wajah 6	4	1	1	2	1.175	40%	20%	91%
wajah 7	4	1	1	2	1.176	63%	25%	92%
wajah 8	4	1	1	1	1.068	47%	15%	91%
wajah 9	4	2	5	7	1.442	35%	60%	87%
wajah 10	4	3	7	10	1.478	26%	65%	81%
wajah 11	3	1	5	6	1.281	16%	27%	84%
wajah 12	2	1	3	4	1.212	24%	30%	90%
wajah 13	1	0	2	3	0.987	0%	0%	93%
Rata-rata					1.235	40%	30%	89%
W. Train	7.445							

LAMPIRAN A (Lanjutan)

A.4 Hasil Klasifikasi wajah SVM RBF

SVM Kelas 10		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	0	0	0	0.680	0%	0%	90%
wajah 2	4	0	0	0	0.560	0%	0%	90%
wajah 3	4	0	0	0	0.608	0%	0%	90%
wajah 4	4	0	0	0	0.742	0%	0%	90%
wajah 5	4	0	0	0	0.563	0%	0%	90%
wajah 6	4	0	0	0	0.735	0%	0%	90%
wajah 7	4	0	0	0	0.651	0%	0%	90%
wajah 8	4	0	0	0	0.641	0%	0%	90%
wajah 9	4	3	29	32	2.009	8%	80%	26%
wajah 10	4	1	7	8	1.233	2%	20%	74%
Rata-rata				40	0.842	1%	10%	82%
W. Train	6.089							

SVM Kelas 11		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	0	0	0	0.747	0%	0%	91%
wajah 2	4	0	0	0	0.615	0%	0%	91%
wajah 3	4	0	0	0	0.736	0%	0%	91%
wajah 4	4	0	0	0	0.883	0%	0%	91%
wajah 5	4	0	0	0	0.649	0%	0%	91%
wajah 6	4	0	0	0	0.797	0%	0%	91%
wajah 7	4	0	0	0	0.694	0%	0%	91%
wajah 8	4	0	0	0	0.724	0%	0%	91%
wajah 9	4	0	0	0	0.720	0%	0%	91%
wajah 10	4	4	39	43	2.900	9%	100%	9%
wajah 11	3	0	0	0	0.653	0%	0%	93%
Rata-rata					0.920	1%	9%	84%
W. Train	6.623							

LAMPIRAN A (Lanjutan)

SVM Kelas 12		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	0	0	0	0.722	0%	0%	91%
wajah 2	4	0	0	0	0.556	0%	0%	91%
wajah 3	4	0	0	0	0.662	0%	0%	91%
wajah 4	4	0	0	0	0.757	0%	0%	91%
wajah 5	4	0	0	0	0.812	0%	0%	91%
wajah 6	4	0	0	0	0.676	0%	0%	91%
wajah 7	4	0	0	0	0.824	0%	0%	91%
wajah 8	4	0	0	0	0.691	0%	0%	91%
wajah 9	4	0	0	0	0.829	0%	0%	91%
wajah 10	4	0	0	0	0.741	0%	0%	91%
wajah 11	3	3	42	45	3.246	7%	100%	8%
wajah 12	2	0	0	0	0.988	40%	20%	96%
Rata-rata					0.959	4%	10%	85%
W. Train	7.308							

SVM Kelas 13		Hasil Pencarian		JMl	Hasil			
	Jml	Benar	Salah		W.Testing	Precision	Recall	Akurasi
wajah 1	4	0	0	0	0.939	0%	0%	91%
wajah 2	4	0	0	0	0.925	0%	0%	91%
wajah 3	4	0	0	0	1.018	0%	0%	91%
wajah 4	4	0	0	0	0.970	0%	0%	91%
wajah 5	4	0	0	0	0.883	0%	0%	91%
wajah 6	4	0	0	0	0.838	0%	0%	91%
wajah 7	4	0	0	0	0.912	0%	0%	91%
wajah 8	4	0	0	0	0.907	0%	0%	91%
wajah 9	4	0	0	0	0.919	0%	0%	91%
wajah 10	4	0	0	0	0.843	0%	0%	91%
wajah 11	3	3	43	46	3.107	7%	100%	7%
wajah 12	2	1	0	1	1.039	40%	20%	97%
wajah 13	1	0	0	0	1.041	0%	0%	98%
Rata-rata				46	1.103	4%	9%	86%
W. Train	8.082							

LAMPIRAN B

B.1 Listing program inisialisasi input dan target serta fungsi SVM

```
jmlDataTraining=2250;
jmlKelas=5;

files = dir('*.jpg');
hist = [];
for n = 1 : 2250
    waitbar(n / jmlDataTraining)

    filename = files(n).name;
    file = imread(filename);

    hist = [hist imhist(rgb2gray(imresize(file,[ 50 50])))];

end

tic;
target=[];
histTrain=hist';
nama=[];
for i=1:jmlKelas-1
    at=1;

    for n = 1 : jmlDataTraining
    %target
         if at>jmlKelas
           at=1;
         end

          ta=at;
          if at<i
              ta=-1;
          elseif at~=i
             ta=0;
          end

          if i==(jmlKelas-1)&&ta==0
              ta=jmlKelas;
          end


         if ta~=-1
             target=[target ta];
             naman=histTrain(n,:);
             nama=[nama naman'];
         end


         at=at+1;
```

```
    end


    %Datatraining
    DataTraining=[nama'  target'];

    %inisialisasi fungsi training
    tic;
    ModelSVM(i) =
       BelajarSVM(DataTraining(:,(1:256)),DataTraining(:,257),i);

    training_time(i) = toc;

target=[];
nama=[];
end


time=sum(training_time);

handles.ModelSVM=ModelSVM;
handles.jmlKelas=jmlKelas;
```

B.2 Listing program training SVM

```
function [svm_struct, svIndex] = BelajarSVM(input, target,
TrainingKe, InkrementKe, varargin)

narginchk(2, Inf);

% cek target vektor apa tidak
if ~isvector(target) && ~ischar(target)
    error(message('stats:svmtrain:GroupNotVector'));
end


if size(target,1) == 1
    target = target';
end

if ~isnumeric(input) || ~ismatrix(input)
    error(message('stats:svmtrain:TrainingBadType'));
end


[targetIndex, targetString] = grp2idx(target);

% cek ukuran data
if size(input,1) ~= size(targetIndex,1)
    if size(input,2) == size(targetIndex,1)
        input = input';
    else
```

LAMPIRAN B (Lanjutan)

```
        error(message('stats:svmtrain:DataGroupSizeMismatch'))
    end
end

if isempty(input)
    error(message('stats:svmtrain:NoData'))
end

nans = isnan(targetIndex) | any(isnan(input),2);
if any(nans)
    input(nans,:) = [];
    targetIndex(nans) = [];
end
if isempty(input)
    error(message('stats:svmtrain:NoData'))
end

ngroups = length(unique(targetIndex));
nPoints = length(targetIndex);

if ngroups > 2
    error(message('stats:svmtrain:TooManyGroups', ngroups))
end
if length(targetString) > ngroups
    warning(message('stats:svmtrain:EmptyGroups'));

end


targetIndex = 1 - (2* (targetIndex-1));

pnames =
[optimMethod, plotflag, polyOrder, mlpParams, boxC,  rbf_sigma,
...
    autoScale, opts, tolkkt, kktvl,kerCL, kfunargs, qpOptsInput,
...
    smoOptsInput] = internal.stats.parseArgs(pnames, dflts,
varargin{:});

%Fungsi Kernel
kfun = @multiKernel;

%fungsi Opimasi
optimList ={'QP','SMO','LS'};
i = 2;

if ~isempty(optimMethod)
    [~,i] =
internal.stats.getParamVal(optimMethod,optimList,'Method');
    if i==1 &&  ( ~license('test', 'optimization_toolbox') ...
            || isempty(which('quadprog')))
        warning(message('stats:svmtrain:NoOptim'));
        i = 2;
    end
end
```

LAMPIRAN B (Lanjutan)

```
if i == 2 && ngroups==1
    error(message('stats:svmtrain:InvalidY'));
end
optimMethod = optimList{i};


qp_opts = optimset('Algorithm','active-set','display','off');
smo_opts = statset('Display','off','MaxIter',15000);

if ~isempty(opts)
    qp_opts = optimset(qp_opts,opts);
    smo_opts = statset(smo_opts,opts);
else

    if ~isempty(qpOptsInput)
        if isstruct(qpOptsInput)
            qp_opts = optimset(qp_opts,qpOptsInput);
        elseif iscell(qpOptsInput)
            qp_opts = optimset(qp_opts,qpOptsInput{:});
        else
            error(message('stats:svmtrain:BadQuadprogOpts'));
        end
    end
end

if ~isempty(smoOptsInput) && isempty(tolkkt) && isempty(kktvl) ...
        && isempty(kerCL) && isempty(opts)

    smo_opts = svmsmoset(smoOptsInput);
else
    if isempty(tolkkt)
        tolkkt = 1e-3;
    end
    if isempty(kerCL)
        kerCL = 5000;
    end
    if isempty(kktvl)
        kktvl = 0;
    end
    smo_opts =
svmsmoset(smo_opts,'tolkkt',tolkkt,'KernelCacheLimit',kerCL,....
        'KKTViolationLevel',kktvl);
end

if ~isscalar(smo_opts.TolKKT) || ~isnumeric(smo_opts.TolKKT) ||
smo_opts.TolKKT <= 0
    error(message('stats:svmtrain:badTolKKT'));
end

if ~isscalar(smo_opts.KKTViolationLevel) ||
~isnumeric(smo_opts.KKTViolationLevel)...
        || smo_opts.KKTViolationLevel < 0 ||
smo_opts.KKTViolationLevel > 1
    error(message('stats:svmtrain:badKKTVL'));
end
```

LAMPIRAN B (Lanjutan)

```
if  ~isscalar(smo_opts.KernelCacheLimit) ||
~isnumeric(smo_opts.KernelCacheLimit)...
        ||smo_opts.KernelCacheLimit < 0
    error(message('stats:svmtrain:badKerCL'));
end


plotflag = bioinfoprivate.opttf(plotflag,'showplot',mfilename);
if plotflag && size(input,2) ~=2
    plotflag = false;
    warning(message('stats:svmtrain:OnlyPlot2D'));
end

if ~isempty(kfunargs) &&  ~iscell(kfunargs)
    kfunargs = {kfunargs};
end


if isscalar(boxC) && isnumeric(boxC) && boxC > 0

    boxconstraint = ones(nPoints, 1);
    n1 = length(find(targetIndex==1));
    n2 = length(find(targetIndex==-1));
    c1 = 0.5 * boxC * nPoints / n1;
    c2 = 0.5 * boxC * nPoints / n2;
    boxconstraint(targetIndex==1) = c1;
    boxconstraint(targetIndex==-1) = c2;
elseif isvector(boxC) && isnumeric(boxC) && all(boxC > 0) &&
(length(boxC) == nPoints)
    % vector input
    boxconstraint = boxC;
else
    error(message('stats:svmtrain:InvalidBoxConstraint'));
end

boxconstraint =
min(boxconstraint,repmat(1/sqrt(eps(class(boxconstraint))),...
    size(boxconstraint)));


% plot the data
if plotflag
    [hAxis,hLines] = svmplotdata(input,targetIndex);
    legend(hLines,cellstr(targetString));
end

if strcmpi(optimMethod, 'SMO')

    if ~isempty(kfunargs)
        tmp_kfun = @(x,y) feval(kfun, x,y, kfunargs{:});
    else
        tmp_kfun = kfun;
    end
```

LAMPIRAN B (Lanjutan)

```
    %optimasi
    [alpha, bias] = seqminopt(input, targetIndex, ...
        boxconstraint, tmp_kfun, smo_opts);

    %Cari support vector
    svIndex = find(alpha > sqrt(eps) & alpha < boxconstraint);
    sv = input(svIndex,:);
    alphaHat = targetIndex(svIndex).*alpha(svIndex);

else

    %hitung fungsi kernel
    try
        kx = feval(kfun,input,input,kfunargs{:});
        % cek simetrik kernel
        kx = (kx+kx')/2 + diag(1./boxconstraint);
    catch ME
        m =
message('stats:svmtrain:KernelFunctionError',func2str(kfun));

throw(addCause(MException(m.Identifier,'%s',getString(m)),ME));
    end

    % Hessian
    H =((targetIndex * targetIndex').*kx);

    if strcmpi(optimMethod, 'QP')
        if strncmpi(qp_opts.Algorithm,'inte',4)
            X0 = [];
        else
            X0= ones(nPoints,1);
        end
        [alpha, ~, exitflag, output] = quadprog(H,-
ones(nPoints,1),[],[],...
            targetIndex',0,zeros(nPoints,1), Inf
*ones(nPoints,1),...
            X0, qp_opts);

        if exitflag <= 0
            error(message('stats:svmtrain:UnsolvableOptimization',
output.message));
        end

        %Cari support vector
        svIndex = find(alpha > sqrt(eps) & alpha < boxconstraint);
        sv = input(svIndex,:);


        %hitung margin
        alphaHat = targetIndex(svIndex).*alpha(svIndex);

        %hitung bias
        [~,maxPos] = max(alpha);
        bias = targetIndex(maxPos) -
sum(alphaHat.*kx(svIndex,maxPos));
```

LAMPIRAN B (Lanjutan)

```
    else
        A = [0 targetIndex';targetIndex,H];
        b = [0;ones(size(targetIndex))];
        x = A\b;

        % hitung parameters garis pemisah dari support vectors.
        sv = input;
        bias = x(1);
        alphaHat = targetIndex.*x(2:end);
        svIndex = (1:nPoints)';
    end
end


 % Save support Vector data
   if nargin <4
    filename = 'SVIndeks.xlsx';
    sheet = TrainingKe;
    xlswrite(filename,svIndex,sheet);

   else
       if InkrementKe==1
            filename = 'SVIndeksInc1.xlsx';
            sheet = TrainingKe;
            xlswrite(filename,svIndex,sheet);
       elseif InkrementKe==2
            filename = 'SVIndeksInc2.xlsx';
            sheet = TrainingKe;
            xlswrite(filename,svIndex,sheet);
       elseif InkrementKe==3
            filename = 'SVIndeksInc3.xlsx';
            sheet = TrainingKe;
            xlswrite(filename,svIndex,sheet);
       end
   end

vm_struct.SupportVectors = sv;
svm_struct.Alpha = alphaHat;
svm_struct.Bias = bias;
svm_struct.KernelFunction = kfun;
svm_struct.KernelFunctionArgs = kfunargs;
svm_struct.target = target;
svm_struct.SupportVectorIndices = svIndex;
svm_struct.FigureHandles = [];
if plotflag
    hSV = svmplotsvs(hAxis,hLines,targetString,svm_struct);
    svm_struct.FigureHandles = {hAxis,hLines,hSV};
end
```

B.3 Listing program Fungsi Kernel

```
function KernelAkhir=multiKernel(nama,nama2)

%linier
```

```
mkl(:,:,1)=(nama*nama2');

%Polynomial
polyOrder=3;
dotP= (nama*nama2');

K=dotP;
for i = 2:polyOrder
    K = K.*(1 + dotP);
end
mkl(:,:,2)=K;

%RBF
rbf_sigma = 1;

mkl(:,:,3) = exp(-
(1/(2*rbf_sigma^2))*(repmat(sqrt(sum(nama.^2,2).^2),1,size(nama2,1
)))...
    -
2*(nama*nama2')+repmat(sqrt(sum(nama2.^2,2)'.^2),size(nama,1),1)))
;

%======SUM=====
Kt=zeros(size(nama,1),size(nama2,1));
Jmlkernel=size(mkl,3);
Dm=1/size(mkl,3);

    for i=1:Jmlkernel
        Kt=Kt+Dm*mkl(:,:,i);
    end
    KernelAkhir=Kt;

end
```

B.4 Listing program Fungsi Testing SVM

```
%Load Gambar
Gambar=handles.Gambar;
GambarRead = imread(Gambar);
histGambar=imhist(rgb2gray(imresize(GambarRead,[ 50 50])));

    A=0;
    k=1;

% Testing
 while A==0&&k<jmlKelas
        A = TestingSVM(ModelSVM(k),histGambar');
        k=k+1;
    end

function outclass = TestingSVM(svmStruct,input, varargin)

% set default
plotflag = false;
```

LAMPIRAN B (Lanjutan)

```
% cek input
narginchk(2, Inf);


if ~isstruct(svmStruct)
    error(message('stats:svmclassify:TwoInputsNoStruct'));
end

if ~isnumeric(input) || ~ismatrix(input)
    error(message('stats:svmclassify:Badinput'));
end

if size(input,2)~=size(svmStruct.SupportVectors,2)
    error(message('stats:svmclassify:TestSizeMismatch'));
end

%
if nargin > 2
    if rem(nargin,2) == 1

error(message('stats:svmclassify:IncorrectNumberOfArguments'));
    end
    okargs = {'showplot','-compilerhelper'};
    for j=1:2:nargin-2
        pname = varargin{j};
        pval = varargin{j+1};
        k = find(strncmpi(pname, okargs,numel(pname)));
        if isempty(k)

error(message('stats:svmclassify:UnknownParameterName', pname));
        elseif length(k)>1

error(message('stats:svmclassify:AmbiguousParameterName', pname));
        else
            switch(k)
                case 1
                    plotflag =
bioinfoprivate.opttf(pval,okargs{k},mfilename);
                case 2
                    svmtrain(eye(2),[1 0]);
            end
        end
    end
end

target = svmStruct.target;

% cek target
if ~isvector(target) && ~ischar(target)
    error(message('stats:svmclassify:GroupNotVector'));
end


[~,groupString,glevels] = grp2idx(target);
```

LAMPIRAN B (Lanjutan)

```
% mengklasifikasi
if ~isempty(input)

    inputOrig = input;
    if ~isempty(svmStruct.ScaleData)
        for c = 1:size(input, 2)
            input(:,c) = svmStruct.ScaleData.scaleFactor(c) * ...
                (input(:,c) +  svmStruct.ScaleData.shift(c));
        end
    end

    try
        outclass = Hasil(input,svmStruct);
    catch ME
        error(message('stats:svmclassify:ClassifyFailed',
ME.message));
    end
    if plotflag

        if isempty(svmStruct.FigureHandles)

warning(message('stats:svmclassify:NoTrainingFigure'));

        else
            try
                hAxis = svmStruct.FigureHandles{1};
                hLines = svmStruct.FigureHandles{2};
                hSV = svmStruct.FigureHandles{3};

                [~,hClassLines] =
svmplotdata(inputOrig,outclass,hAxis);
                trainingString = strcat(cellstr(groupString),'
(training)');
                inputString = strcat(cellstr(groupString),'
(classified)');

legend([hLines(1),hClassLines(1),hLines(2),hClassLines(2),hSV],...
                    {trainingString{1},inputString{1},...
                    trainingString{2},inputString{2},'Support
Vectors'});
            catch ME
                warning(message('stats:svmclassify:DisplayFailed',
ME.message));
            end
        end
    end
    outclass(outclass == -1) = 2;
    unClassified = isnan(outclass);
    outclass = glevels(outclass(~unClassified),:);
    if any(unClassified)

         try
             outclass =
dfswitchyard('statinsertnan',unClassified,outclass);
         catch ME
```

```
            if
~isequal(ME.identifier,'stats:statinsertnan:LogicalInput')
                rethrow(ME);
            else

error(message('stats:svmclassify:logicalwithNaN'));
            end
        end
    end

else
    outclass = [];
end

% Fungsi Keputusan
function [out,f] = Hasil(Xnew,svm_struct)

sv = svm_struct.SupportVectors;
alphaHat = svm_struct.Alpha;
bias = svm_struct.Bias;
kfun = svm_struct.KernelFunction;
kfunargs = svm_struct.KernelFunctionArgs;


f = (feval(kfun,sv,Xnew,kfunargs{:})'*alphaHat(:)) + bias;
out = sign(f);

out(out==0) = 1;
```

B.5 Listing Program Proses Penambahan Kelas

```
h = waitbar(0,'Please Wait...');
jmlKelas=handles.jmlKelas;

if get(handles.text12, 'String')=='0'
% ========================
        %Load Gambar baru Increment 1
        files = dir('*.jpg');
        hist = [];
        target=[];
        jmlDataTraining=225;
        Kelas=jmlKelas+1;

        %training
        for n = 2726 : 2750
            waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
```

LAMPIRAN B (Lanjutan)

```
        end

        for n = 2501 : 2700
            waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
        end

        Data1=[hist'  target'];
        set(handles.text12, 'String', '1');

elseif get(handles.text12, 'String')=='1'
% ========================
        files = dir('*.jpg');
        hist = [];
        target=[];
        jmlDataTraining=135;
        Kelas=jmlKelas+1;

        %training
        for n = 2886 : 2900
             waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
        end

        for n = 2751 : 2870
             waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
        end
        Data2=[hist'  target'];
        set(handles.text12, 'String', '2');

elseif get(handles.text12, 'String')=='2'
% ========================

        files = dir('*.jpg');
        hist = [];
        target=[];
```

LAMPIRAN B (Lanjutan)

```
        jmlDataTraining=90;
        Kelas=jmlKelas+1;

        %training
        for n = 2991 : 3000
             waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
        end

        for n = 2901 : 2980
             waitbar(n / jmlDataTraining)

            filename = files(n).name;
            file = imread(filename);

            hist = [hist imhist(rgb2gray(imresize(file,[ 50
50])))];
            target= [target Kelas];
        end
        Data3=[hist'  target'];
        set(handles.text12, 'String', '3');

end


InkrementKe=str2num(get(handles.text12, 'String'));
for i=1: jmlKelas

    %Load Data
    if get(handles.text12, 'String')=='1'

        if i<jmlKelas
            filename = 'Trainingdata.xlsx';
            sheet = i;
            xlRange = 'A1:IW3000';
            LoadDT = xlsread(filename, sheet, xlRange);

            filename = 'SVIndeks.xlsx';
            sheet = i;
            xlRange = 'A:A';
            LoadSVindeks = xlsread(filename, sheet, xlRange);

            DataSV=LoadDT(LoadSVindeks,:);
            DTrain=[DataSV;Data1];

            zz=size(DTrain);
            for j=1:zz(1,1)
                if DTrain(j,257)~=i
                    DTrain(j,257)=0;
```

LAMPIRAN B (Lanjutan)

```
            end
        end

    else

        filename = 'Trainingdata.xlsx';
        sheet = (i-1);
        xlRange = 'A1:IW3000';
        LoadDT = xlsread(filename, sheet, xlRange);

        filename = 'SVIndeks.xlsx';
        sheet = (i-1);
        xlRange = 'A:A';
        LoadSVindeks = xlsread(filename, sheet, xlRange);

        DataSVPra=LoadDT(LoadSVindeks,:);
        zz=size(DataSVPra);

        DataSV=[];
        for j=1:zz(1,1)
            if DataSVPra(j,257)==i
                DataSV=[DataSV DataSVPra(j,:)'];
            end
        end
        DataSV=DataSV';
        DTrain=[DataSV;Data1];

    end

    filename = 'TDIncrement1.xlsx';
    sheet = i;
    xlswrite(filename,DTrain,sheet);

elseif get(handles.text12, 'String')=='2'

    if i<jmlKelas
        filename = 'TDIncrement1.xlsx';
        sheet = i;
        xlRange = 'A1:IW3000';
        LoadDT = xlsread(filename, sheet, xlRange);

        filename = 'SVIndeksInc1.xlsx';
        sheet = i;
        xlRange = 'A:A';
        LoadSVindeks = xlsread(filename, sheet, xlRange);

        DataSV=LoadDT(LoadSVindeks,:);
        DTrain=[DataSV;Data2];

        zz=size(DTrain);
        for j=1:zz(1,1)
            if DTrain(j,257)~=i
                DTrain(j,257)=0;
            end
        end
```

LAMPIRAN B (Lanjutan)

```
        else

          filename = 'TDIncrement1.xlsx';
          sheet = (i-1);
          xlRange = 'A1:IW3000';
          LoadDT = xlsread(filename, sheet, xlRange);

          filename = 'SVIndeksInc1.xlsx';
          sheet = (i-1);
          xlRange = 'A:A';
          LoadSVindeks = xlsread(filename, sheet, xlRange);

          DataSVPra=LoadDT(LoadSVindeks,:);
          zz=size(DataSVPra);

          DataSV=[];
          for j=1:zz(1,1)
              if DataSVPra(j,257)==i
                  DataSV=[DataSV DataSVPra(j,:)'];
              end
          end
          DataSV=DataSV';
          DTrain=[DataSV;Data2];


      end

      filename = 'TDIncrement2.xlsx';
      sheet = i;
      xlswrite(filename,DTrain,sheet);

  elseif get(handles.text12, 'String')=='3'

      if i<jmlKelas
          filename = 'TDIncrement2.xlsx';
          sheet = i;
          xlRange = 'A1:IW3000';
          LoadDT = xlsread(filename, sheet, xlRange);

          filename = 'SVIndeksInc2.xlsx';
          sheet = i;
          xlRange = 'A:A';
          LoadSVindeks = xlsread(filename, sheet, xlRange);

          DataSV=LoadDT(LoadSVindeks,:);
          DTrain=[DataSV;Data3];

          zz=size(DTrain);
          for j=1:zz(1,1)
              if DTrain(j,257)~=i
                  DTrain(j,257)=0;
              end
          end
```

LAMPIRAN B (Lanjutan)

```
          else

            filename = 'TDIncrement2.xlsx';
            sheet = (i-1);
            xlRange = 'A1:IW3000';
            LoadDT = xlsread(filename, sheet, xlRange);

            filename = 'SVIndeksInc2.xlsx';
            sheet = (i-1);
            xlRange = 'A:A';
            LoadSVindeks = xlsread(filename, sheet, xlRange);

            DataSVPra=LoadDT(LoadSVindeks,:);
            zz=size(DataSVPra);

            DataSV=[];
            for j=1:zz(1,1)
                if DataSVPra(j,257)==i
                    DataSV=[DataSV DataSVPra(j,:)'];
                end
            end
            DataSV=DataSV';
            DTrain=[DataSV;Data3];


        end

        filename = 'TDIncrement3.xlsx';
        sheet = i;
        xlswrite(filename,DTrain,sheet);

    end

    tic;
    ModelSVM(i) =
BelajarSVM(DTrain(:,(1:256)),DTrain(:,257),i,(InkrementKe));
    training_time(i) = toc;

end

time=sum(training_time);
jmlKelas=jmlKelas+1;

handles.ModelSVM=ModelSVM;
handles.jmlKelas=jmlKelas;
```

BIODATA PENULIS

Penulis dilahirkan di Sidoarjo, 05 Desember 1990, merupakan anak pertama dari tiga bersaudara dari pasangan Bapak Yusup dan Ibu Lu'aiyah. Penulis sekarang bertempat tinggal di Desa Kepatihan RT/RW 02/02, Tulangan, Sidoarjo. Penulis telah menempuh pendidikan formal, yaitu di SDN Kepatihan 1, SMPN 1 Tulangan, dan SMAN 3 Sidoarjo. Setelah lulus tahun 2009, penulis diterima di S-1 Jurusan Matematika Institut Teknologi Sepuluh Nopember (ITS) melalui jalur PMDK Regular dan terdaftar dengan NRP 1209 100 015. Setelah lulus dan mendapatkan gelar sarjana saint pada tahun 2013, penulis melanjutkan studi S-2 di Jurusan Matematika ITS Surabaya melalui beasiswa Fresh Graduate di tahun yang sama dan terdaftar dengan NRP 1213 201 056. Selama kuliah S-1 dan S-2 di jurusan matematika, penulis mengambil bidang ilmu komputer. Kritik dan saran ataupun pertanyaan yang berhubungan dengan tesis ini dapat menghubungi penulis melalui email.

Email : athoillah.muhammad@gmail.com